Department of Computer Science

Technical Report

Undergraduate Dissertation: Development of an Integrated Chat Monitoring and Web Filtering Parental Control for Child Online Supervision

Kevin Chung

Technical Report 2004-13 **May 2004**

ISSN 1740-9497

Contact Address:

Department of Computer Science
University of Bath
Bath, BA2 7AY
United Kingdom

URL: http://www.cs.bath.ac.uk

ISSN 1740-9497

Development of an Integrated Chat Monitoring and Web Filtering Parental Control for Child Online Supervision

Kevin Chung

BSc (Hons) Computer Science

Univeristy of Bath

May 2004

……..

Development of an Integrated Chat Monitoring and Web Filtering Parental Control for Child Online Supervision

submitted by Kevin Chung

DECLARATION

This dissertation is submitted to the University of Bath in accordance with the requirements of the degree of Bachelor of Science in the Department of Computer Science. No portion of the work in this dissertation has been submitted in support of an application for any other degree or qualification of this or any other university or institution of learning. Except where specifically acknowledged, it is the work of the author.

……..

Abstract

Advancements in computer and telecommunication technology in recent years has resulted in the increasing use of computers and the Internet, allowing children to access vast quantities of information and knowledge on the Internet, communicate with friends and family though email, or even real-time communication through the use of instant messaging or chat rooms more than ever before.

Along with this growth comes an amplified requirement for security – parents need reassurance that their children will be safe from accessing inappropriate material such as pornography and more importantly from potential exploitation and harm by computer-sex offenders. Although parents may educate their children on 'safe' Internet practise and occasionally accompany them whilst they are online, it is neither practical nor convenient to give supervision to a child for 100% of the time that they are online, and a child's 'online behaviour' may also be different when not under the watchful eye of a parent.

The aim of this project is to research and develop an application, which will allow parents to monitor a child's Internet activities without giving 'over-the-shoulder' supervision. The application will enable filtering of what content a child has access to, and also provide alerts as and when a child views inappropriate sites or partakes in inappropriate conversation.

With Special Thanks To

Professor James Davenport for his guidance and supervision

And to my friends and family for their love and support

Contents

1 Introduction

1.1 Background

In June 1997, there were an estimated 1 million users online in the UK. In just 5 years this has ballooned into over 34.4 million[1] users, representing approximately 57% of the total UK population (URL-NUA, 2003). Of these, there are an estimated 4.8 million children online, of whom over 1 million are under 14. 65% of all 7 to 16 year olds in the UK have used the Web and are frequent users of email to communicate with friends, family and virtual friends. 62% of these children used the Internet at home, and in addition 81% of all young users have access from school (Dixon et al. 2003). This extraordinary increase in numbers has led to a new cyber-society, where members interact in cyberspace just as they would in the real world. However, as with any society, there also exists the 'anti-social' element and cyberspace is no different.

> *"We were all taught about not talking to strangers by our parents but there are few people who are computer savvy enough to pass on advice to kids who use the Internet. Today's children are the first generation to be brought up using computers in their everyday lives, and, there is an information gap that needs to be filled." – NCH Spokesman, January 2003.* [2]

These are the chilling words that bring to light the strides in advancement achieved in recent years in terms of computer and information technology. With the development of faster computers and faster Internet connections, a new world of information and exploration is revealed, expanding horizons and exposing us to new culture, communication and creative expression. As such, the usual 'parental' advice and actions filtered down through the generations, hasn't yet been developed, in terms of how parents should advise on and supervise the online activities of their children. Of course, this will eventually be developed and the 'information gap' filled, but in the immediate future there is a very real, unprecedented danger to children from the criminal, anti-social element of the online society. Uneducated and unmonitored, children could be open to a world of harmful physical and psychological risks such as being approached online by persons with the aim of developing sexual relations with them, which may incorporate other risks including:

- Children being exposed to inappropriate conversation;
- Children unwittingly becoming the subject of sexual fantasy;
- Children being sent indecent or obscene images;
- Children being asked to send indecent images of themselves and/or their friends;
- Children being engaged in explicit sexual talk and and/or being encouraged to perform sexual acts on themselves and/or their friends (so-called 'cybersex'). (Dixon et al. 2003)

These risks although small, are very real, highlighted in at least 12 cases in recent years [3] of children being physically attacked by men they have met online.

---

[1] Survey date September 2002.
[2] http://news.bbc.co.uk/1/hi/uk/2631025.stm (Accessed 25/11/03)
[3] http://news.bbc.co.uk/1/hi/technology/2632197.stm (Accessed 25/11/03).

1.2 Motivation

The motivation for this project stems from the authors' interest in network and computer security, subsequent to working in this field during industrial placement, and in conjunction to the opportunity to design and program a network application that would involve security aspects. The idea to focus this interest towards developing a child Internet supervisor developed from widely publicised cases in the media concerning increases in cases of child victimisation and abduction following online acquaintances with paedophiles posing as teenagers. Initial investigation raised concerns that many parents are not aware of the potential risks involved with unrestricted Internet access, or the precautions that can be taken. As a consequence, the objectives of this project are to bring awareness to the aforementioned issues, and to design and develop an application that will aid in addressing these issues.

1.3 Document Overview

This document begins by examining in detail, the background behind the need for a child Internet monitor, including the dangers involved on the Internet, the ethical issues and legislation surrounding such software, and finally a high-level overview of the current solutions available in Chapter 2.

Chapter 3 extends the overview of currently available solutions by analysing several commercial products which focus on tackling the problems of protecting children online in detail, discussing their relative strengths and weaknesses.

Requirements analysis and specification is covered in Chapter 4, discussing the main areas of focus for the application, design issues and constraints.

Chapter 5 details the design methodology, the analysis of MSN Messenger protocol, the high-level design of the analysing engine, and the design of the graphical user interface required of the administration console. Chapter 6 extends and discusses the implementation of the application in greater detail.

Chapter 7 and Chapter 8 will discuss application-testing and concluding remarks respectively.

2 Literature Review

The focus of the literature review will concentrate on three main sections examining the background of the identified issues in more depth:

1. Dangers on the Internet including pornography and chat rooms.
2. The ethical and legislative issues surrounding access control software, and governmental guidelines for Internet use.
3. High-level overview of current technological solutions.

Closer analysis of existing solutions is examined in Chapter 3, where the discussion of their relative strengths and weaknesses can be discussed in further depth than required in the literature review.

2.1 Online Dangers

A recent study by Lancaster University (URL-LAN, 2002)[4], suggests that 1 in 5 children who use the Internet used chat rooms, and of those, 1 in 10 have met in person somebody they communicated with online. More disturbingly, an adult did not accompany 3 in 4 of these children who went to face-to-face meetings. 1 in 2 children reported that their parents never supervised their online activities, and 1 in 3 children did not know where to report unpleasant experiences and would not have told parents.

There have been many recent publications in the media of several arrests, including high-profile celebrities who visit sites and download indecent pictures, and also paedophiles that use chat rooms to lure children into offline meetings. The victims of this type of abuse are usually tricked into these meetings by building up a false sense of security with the offender through online conversation (Shoniregun and Anderson, 2003). More detail of some of these publications can be viewed in the 'Case Studies ' section of this section.

2.1.1 Chat Rooms: Exposure and engagement in inappropriate conversation

A chat room can be defined as a place on the Internet where people communicate by typing messages. The text is displayed almost instantly on the computer screens of everyone else in the chat room, wherever they are in the world. Someone a person meets in a chat room might become one of their 'friends' even though they have never met in the real world (URL-WTN, 2003). Examples include: Yahoo! Chat [5], Excite – Chat [6], and until recently in the UK, MSN Chat. Due to the problems that are explained later in this section, Microsoft decided to close the vast majority of its chat rooms in September 2003 (Carter, 2003).

[4] From a sample of 1369.
[5] http://chat.yahoo.com/ (Accessed 5/12/03)
[6] http://communicate.excite.com/chat.html?PG=home&SEC=feat (Accessed 5/12/03)

Another form of online chat includes 'Instant Messaging' which allows instant message's to be sent to a friend on the Internet in much the same way as you can send a text (written) message to their mobile phone. Both parties must agree to receive messages from each other to use the service, and it is an easy and useful way of keeping in touch with friends. Unlike chat rooms, Instant Messaging is usually more closely associated with a network of friends, as parties must know the 'sign-in' name of each other in order to exchange messages (URL-WTN, 2003). Examples include MSN Messenger [7], ICQ [8] and AOL Instant Messenger [9].

2.1.1.1 Problems

For many paedophiles, the Internet has become the means of not only finding pornographic images, but also potential victims. Known as "grooming", the process usually involves the offender making contact with children in chat rooms. Once contact is made in a chat room, it can escalate very quickly to mobile phone calls, text messaging and, eventually, face-to-face contact. Over a period of weeks or months, he will try to befriend a youngster - particularly someone who appears naive or vulnerable. What begins as a seemingly innocent online conversation eventually turns to sex, as the paedophile tries to snare his victim. Typically, the child may be encouraged to take indecent photographs, and then email the images to their new "friend". Eventually, the paedophile may try to persuade the victim to agree to a meeting (Gould, 2002)

2.1.1.2 Case Studies

The following reports from national publications highlight the dangers given above:

1. October 2000. Patrick Green, 33, Bucks, was jailed for 5 years after luring a 13-year-old girl to his home for sex after meeting her through an Internet chat room in what is believed to be the first prosecution of its kind in Britain (Pook, S. The Daily Telegraph. 2000)

2. June 2003. Michael Wheeler, 36, of Cambridge, was jailed for three years after pleading guilty to 11 sex offences against young girls. He had abused two 13-year-olds after making contact with them via a chat room (Carter, H. The Guardian. 2003)

3. July 2003. A former US marine has been charged with abducting a 12-year-old girl who he met in an Internet chat room. Toby Studabaker, 31, from Michigan, sparked an international manhunt when he and the youngster disappeared. He said when arrested that he had thought she was older. (Ward, D. The Guardian. 2003)

4. September 2003. A 15-year-old schoolgirl was allegedly raped in Wigan by a man whom she had met through an Internet chat room. The teenager was tricked into meeting the man after he claimed to be a teenager (Carter, H. The Guardian. 2003)

[7] http://messenger.msn.com/ (Accessed 5/12/03)
[8] http://web.icq.com/ (Accessed 5/12/03)
[9] http://www.aol.com/community/chat/allchats.html (Accessed 5/12/03)

2.1.2 Content: Pornography and other undesirable material

Fact: Pornography is big business. 12% of Internet sites are pornographic and between them they boast a US$58 Billion annual turnover (URL-POL, 2003).

The common practice of today's Internet 'pornopreneur' is the posting of free teaser images on their websites as enticements to solicit new subscribers. Spam email, links from other web pages, and pop-up windows are all ways in which the 'pornopreneurs' can bring attention to their website. Any child with unrestricted Internet access can view these free pictures through accidentally accessing such sites or by deliberately searching them out. Hughes (2001), points out that there are many ways in which a child can accidentally come into contact with pornography including:

- Innocent, imprecise, misdirected searches: pornopreneurs' often use popular terms in a bid to increase the traffic to their sites. When children key in their favourite search terms, pornographic sites often pop up along with the sites the children are seeking. Examples include searching for words such as toys, boys, Britney Spears and dogs.

- Stealth sites and misleading URLs: children seeking information on the White House in the US, may find themselves on a porn site instead of the official site at www.whitehouse.gov. Pornographers often purchase domain names such as the .com equivalent of a popular .gov or .org website, knowing full well that web surfers are likely to end up on their pornographic site instead of their desired destination.

- The misuse of brand names:

    - According to a recent study in England, 26 popular children's characters, such as Pokemon, My Little Pony and Action Man, revealed thousands of links to porn sites. 30% of the sites were hard-core. (Envisional, 2000, referenced by Hughes)

    - 25% of porn sites are estimated to misuse popular brand names in search engine magnets, metatags and links. Three of the top ten brand names used are specifically targeted to children - Disney, Barbie, and Nintendo. (Cyveillance Survey, 1999, referenced by Hughes)

Other than pornography sites, there are millions of other inappropriate sites containing unsuitable information that exist on the web. These include sites which contain violent or hateful material, or which advocate the use of weapons or harmful substances such as alcohol, tobacco, or illegal drugs.

2.1.2.1 Problems

The problems of children accessing information such as that described above are obvious – highlighted by the fact that in the UK many high street newsagents do not sell any form of hard core pornography if any at all. For newsagents who do, many precautions are taken to ensure that the material in question is placed on the highest bookshelf so that it is out-of-reach of children, and also checks are made to ensure that the person buying the material is in fact over the age of 18. Information containing hate and violent material is also moderated (for

example films are given a rating from U – suitable for all, up to 18 – suitable for persons aged 18 or over only) so to prevent children from obtaining material that is deemed too violent or explicit in nature.

On the Internet however, such precautions are rarely in place and as Hughes mentions above, a child can very easily stumble across pornographic and inappropriate material. Because of the amount of, and ease at which a child can access pornographic material on the Internet, I have also picked out and highlighted some very disturbing points that are mentioned by Hughes (2001) in her report:

- According to one study, early exposure (under fourteen years of age) to pornography is related to greater involvement in deviant sexual practice, particularly rape. Slightly more than one-third of the child molesters and rapists in this study claimed to have at least occasionally been incited to commit an offence by exposure to pornography. Among the child molesters incited, the study reported that 53 percent of them deliberately used the stimuli of pornography as they prepared to offend.[10]

- The habitual consumption of pornography can result in a diminished satisfaction with mild forms of pornography and a correspondingly strong desire for more deviant and violent material. [11]

- Children often imitate what they've seen, read, or heard. Studies suggest that exposure to pornography can prompt kids to act out sexually against younger, smaller, and more vulnerable children. Experts in the field of childhood sexual abuse report that any premature sexual activity in children always suggests two possible stimulants: experience and exposure. This means that the sexually deviant child may have been molested or simply exposed to sexuality through pornography.[12]

These points are visibly upsetting and harrowing showing that the threat of paedophiles on the Internet is clearly not the only issue that parents should be worried about. Along, with the danger of 'normal' porn, there is also a growing problem of pornographic material of children appearing on the web.

> *Detective Inspector Terry Jones, of Greater Manchester police:*
>
> *"About seven years ago, we started to see a massive rise in the number of images of children. In 1995, my unit received 12 still and video images from Greater Manchester. This kind of thing was rare then. But in 1999, we received 41,000 images, all of them bar three on computers. We don't count now. We found one man in Manchester who had 50,000 images himself.*
>
> *Some pictures were accompanied by sound on which children could be heard crying and screaming. Virtually all these images are of children being abused. We must never lose sight of the fact that, when we look at an image, some poor kid somewhere has been, and still may be, at risk." (Ward, D. The Guardian. 2002)*

[10] W. L. Marshall, "The Use of Sexually Explicit Stimuli by Rapists, Child Molesters, and Nonoffenders," *The Journal of Sex Research 25*, (Referenced by Hughes)
[11] D. Zillmann, "Effects of Prolonged Consumption of Pornography," in *Pornography: Research Advances and Policy Considerations (*(Referenced by Hughes)
[12] Stephen J. Kavanagh, *Protecting Children in Cyberspace* (Referenced by Hughes)

Reiterating the fact pointed out at the beginning of this subsection, 12% of all Internet sites are pornographic in nature, and without a doubt, this figure increases with everyday that passes.

2.1.2.2 Case Studies

The following reports from national publications highlight the dangers given above:

- April 2002. Police from 34 forces raided more than 70 homes and work places across the country yesterday in one of Britain's biggest ever operations against paedophiles that use Internet chat rooms to swap pornographic images of children. 27 people were arrested in raids coordinated by detectives in Hertfordshire and Greater Manchester. Police said those involved were "not your stereotypical dirty old man in a Mac" and included a 15-year-old boy from Cleveland, a 17-year-old from Hertfordshire and a teacher from south Wales (Ward, D. The Guardian. 2002).

- November 2001. Up to 130 suspected paedophiles in 19 countries were arrested yesterday in a co-ordinated national operation against those downloading child pornography from the Internet. The results were the culmination of a 10-month operation, Landmark, which the NCS said was the world's largest collaborative policing operation. It focused on newsgroups that suspects used to “request, exchange and supply paedophilic images”. Police retrieved more than 100,000 images of children. One suspect asked other newsgroup visitors for advice on "grooming" a child for sex (Steele, The Daily Telegraph. 2001).

2.2 Ethics

Despite the development in technology and applications associated with the Internet, the question of how to legislate the Internet has made minute movement in comparison to its growth, partially due to the question of ownership. The Internet does not ‘belong’ to any governing body, and as a result there are struggles to control the content that it provides (Shoniregun and Anderson, 2003), given that legislation in one country cannot be applied to a service that runs in another country. However, work has begun on both sides of the Atlantic, with governments introducing new legislation that will hopefully make cyberspace a safer place for children.

The current high profile issues presented by online chat rooms and inappropriate content has resulted in an recent escalation of activity in new government legislation and advice, designed to protect the younger generation and curb individuals on the Internet who could otherwise exploit the present vulnerabilities in the system. To illustrate what each individual country has implemented is way beyond the scope of this paper. Instead, focus will be on the recent actions taken by the British government.

2.2.1 Government Legislation

Currently in the UK, several existing laws can be used to cover offences committed through chat room activity consisting of or directed towards inappropriate sexual communication or contact with a child.

The Obscene Publications Act (1959), Protection of Children Act (1978), Criminal Justice Act (1984), Telecommunications Act (1984) and Malicious Communications Act (1988) can be applied to the sending of obscene or distressing communications to a minor in a chat room. Offline offences can also be dealt with under the Indecency with Children Act (1960), Sexual Offences Act (1956), Child Abduction Act (1984), Sexual Offences (Conspiracy and Incitement) Act (1996) and Criminal Justice (Terrorism and Conspiracy) Act (1998), depending on the exact circumstances (Dixon, 2001).

However, concerns over whether these current laws were effective enough in today's environment resulted in the government proposing changes in 2000 to the Sexual Offences Act, which would provide legislation for more stringent penalties to deal with concerns over online enticement. This was eventually passed through Parliament in November 2003, ready for implementation in May 2004 (URL-SOB).

The new changes include:

- A serious new offence of adult sexual abuse of a child, applicable to all adult (over 18) sexual acts with a child under the age of consent (16);
- New offences to bear heavily on those who commercially exploit children including recruiting, inducing or compelling a child.
- The law setting out specific offences against children should state that below the age of 13, a child cannot effectively consent to sexual activity.
- The introduction of abuse of trust offences for adults who are in certain positions of trust or authority over a child

However, concerns remain that these changes mean that the law remains 'reactive' rather than 'proactive'. Dixon et al. suggest that a specific proposal should be made so that it should be possible to criminalize the intent (namely 'culpable misrepresentation to a minor' rather than the positive actions of the suspect), giving the police greater powers to act preventatively instead of having to wait for the actual offence to be committed. The implication of this is, although it acts as a deterrent, there will always be perpetrators in society who are happy to break these laws, until they are caught 'red-handed'.

2.2.2 Guidelines

In general, the following guidelines have been provided for parents on educating and monitoring their children's Internet use by several sources including the British Computer Society (URL-BCS), the Internet Watch Foundation (URL-IWF), and the Federal Bureau of Investigation (URL-FBI):

2.2.2.1 For Parents

- Spend time with your children on-line. Have them teach you about their favourite on-line destinations.

- Keep the computer in a common room in the house, not in your child's bedroom. It is much more difficult for a computer-sex offender to communicate with a child when the computer screen is visible to a parent or another member of the household. It will also prevent a child from becoming a 'loner' fixated on computer games or chat rooms etc.

- Communicate, and talk to your child about sexual victimisation and potential on-line danger.

- Teach your child the responsible use of the resources on-line. There is much more to the on-line experience than chat rooms.

- Understand, even if your child was a willing participant in any form of sexual exploitation that he/she is not at fault and is the victim. The offender always bears the complete responsibility for his or her actions.

- Report incidents if you feel that your child does receive any inappropriate information or engage in inappropriate activities online.

- Parents should educate their children about 'SMART' Internet browsing (detailed in section 3.2.2).

- Utilise parental controls provided by your service provider and/or blocking software. Use of chat rooms, in particular, should be heavily monitored.

- Always maintain access to your child's on-line account and randomly check his/her e-mail. Be up front with your child about your access and reasons why.

- Finally, parents should set up an 'online' contract for acceptable terms of Internet use. An example can be found at: http://www.icra.org/_en/kids/familycontract/ (Accessed 5/12/03).

2.2.2.2 For children

Adhere to the SMART rules whilst surfing the Internet:

- Keep your personal details **Secret.** Don't give out personal details, photographs, or any other information that could be used to identify you, such as information about your family, where you live or the school you go to.

- Never **Meet** someone you have contacted in Cyberspace without your parent's/carer's permission, and even then only when they can be present. The first meeting should always be in a public place. Always stay in the public areas of chat where there are other people around.

- Don't **Accept** e-mails, open attachments or download files from people or organisations you don't know or trust - they may contain viruses or nasty images or messages.

- **Remember** that someone online may not be who they say they are. If you feel uncomfortable or worried in a chat room simply get out of there! Don't take other people at face value - they may not be what they seem, and what they tell you online may or may not be true.

- **Tell** your parent or carer if someone or something makes you feel uncomfortable or worried. Never respond directly to anything you find disturbing - save or print it, log off, and tell an adult.

2.2.3 Education or Big Brother?

Opinions about the ethics of surveillance vary widely from person to person, and also from country to country. The European Union considers the monitoring of employees in the work place as a human rights violation, particularly when it's done surreptitiously (Glass, 2002). However, in both the UK and the US, monitoring is generally the prerogative of an employer, as long as monitoring policies are uniformly implemented, and employees warned that the system is being monitored. Hirsh (2001) states that recent research shows that 77% of all US companies record and review employee communications including email, Internet connections and computer files. This is to ensure that time is not wasted on the Internet for personal use, and that libellous emails are not sent out under the corporate banner.

However, at home, the main dilemma is neither of wasted corporate time nor the danger of bringing the company into disrepute, but the big problem of online anonymity in that no one can ever be 100% sure of who they're talking to online. On one hand, children have a right to be treated with dignity and respect, extending to respect of their privacy both in the real as well as online world as well. It is through this respect that allows a parent to encourage a child to make his/her own decisions in life that ultimate allows them to mature through life. However, on the other hand, parents also have a responsibility to make sure kids are safe. With the dangers highlighted in this document, it is evident that parents must be involved when it comes to education and monitoring their children on the Internet.

Interestingly, in the guidelines given by the afore-mentioned institutes, the majority advocate the use of filtering and monitoring software to:

- Filter and screen out undesirable material depending on the age of the child
- Monitor the people that the child communicates with through the Internet

This is because it is not possible, technically or practically, to supervise and monitor every web page browsed, and every online conversation a child makes on the Internet.

Also in support of using monitoring software, Willard (2002) suggests that the existence of effective supervision and monitoring in schools — with student knowledge of such existence — is generally a sufficient deterrent for misuse:

> *"When students are fully aware that there is a high probability that instances of misuse will be detected and result in disciplinary action, they are unlikely to take the risk of engaging in such misuse. This is equivalent to the standards for inspection of desks and*

lockers, except that the monitoring is occurring at all times. Special inspection of the online activities of an individual student would occur only when the monitoring system detects activity that raises a reasonable suspicion that inappropriate activity has occurred. Students should also be reminded that their parents also have the right to inspect their desks or lockers upon request. The same standard should be applied to Internet use records."

Finally, Zitz (2003) probably makes the most important point of all in his article:

> *"Although you may feel as though it is spying, checking what they have been doing online may avoid further problems, by talking to them about what they have seen. Perhaps they ran across something that made them feel uncomfortable and they feel uncomfortable telling you about it."*

Overall, although there is the moral dilemma that children should be allowed to develop on their own through trust, respect, and education, it seems that the general consensus is one that is moving towards the use of 'Big Brother' monitoring software to ensure that children are safe when online on the Internet, on top of education of how to use the Internet correctly. Reiterating a point made earlier, the current 'information gap' caused by the rapid expansion and development of technology has resulted in parents lacking knowledge of the contents that the Internet poses to children. This problem is currently being addressed by the UK government, through a series of advertising campaigns informing parents to 'Wise Up To The Net' (URL-WTN). Perhaps we are now seeing the 'beginning' of regulation of the Internet in terms of how access should be restricted for different age groups, just like the way a central governing body rates films depending on content, and parents then given the decision of whether or not to allow their children to view it.

2.3 Current Approaches

Currently, a wide variety of technology-based tools for assisting in Internet safety are available on the market to parents, educational institutions and business organisations. These typically fall into 2 main categories: 'Filtering' - which allows the instigation of preventative measures to block access to inappropriate sites, and 'Monitoring' – which allows parents to view logs and activities of the child's online activities, allowing reactive action to be taken if evidence is shown that the child is attempting to reach inappropriate sites, or is engaging in inappropriate conversation.

2.3.1 Filtering

Software filters can be implemented either on the server-side, that is, on the computer that provides the service such as the school Internet gateway, the search engine, the chat room host, and even the ISP (Internet Service Provider) itself, or, client-side, which are filters implemented on the actual computer the child is working on, allowing a parent to granularly define what is filtered.

Most software filters use some combination of the following three strategies for limiting access to websites:

1. Block Lists
2. Word Recognition
3. Website Ratings

2.3.1.1 Block List

A block list is the simplest filtering method, typically involving building and maintaining a list of forbidden sites, usually based on DNS name, or IP address. While some software companies pick what should be filtered, others let parents do it, among pre-set categories, allowing further sites to be blocked when deemed necessary.

The big problem about this method is that it is impossible to keep up with the thousands of websites that are being created every day, and keeping lists up-to-date is a tedious and laborious process. Usually, software companies operate by employing people to constantly update these lists, by looking at and classifying web pages into categories that a parent may or may not choose to block. The parent using the software can then periodically download these new updated lists.

The filtering method described above is known as 'black listing' as everything is allowed apart from sites on the 'black list'. The opposite of this is to block everything and only allow pages that are deemed as developmentally appropriate, educational and entertaining to be viewed. This approach is known as 'white listing', and negates the problem of keeping the black list up-to-date. However, the drawback of this method is that it will restrict vast amounts of information on the Internet – something that makes it so useful in the first place.

2.3.1.2 Word Recognition

Word recognition filtering involves scanning web pages for certain objectionable words or phrases. This technology can be used both on client-side computers which scans the web pages, blocking websites containing these character strings, and also on server-side computers, for example on chat room servers to recognise and block particular character strings which contravene the acceptable use policy to be displayed.

There are however 2 main problems with this system. The first problem is that word recognition filtering may filter out pages with educational material because it recognises words which although on the block list may have different meanings in different contexts. For example, a child may be unable to research a topic such as 'breast cancer' because the word 'breast' is deemed inappropriate and as such any web page with a reference to that word is blocked. The second problem is that although it may filter out objectionable words and phrases such as sex, porn, drugs etc, these phrases are all in English, and as such it is still possible for pages with this content to be displayed in other languages.

2.3.1.3 Website Ratings

The final filtering method is to filter sites so that children can only access sites with the appropriate rating through the web browser. This form of filtering is already implemented in major web browsers such as Internet Explorer, using the RSACi (Recreational Software

Advisory Council) (URL-RSAC) rating system to allow parents to specify the level of nudity, sex and violence permitted.

However, this system relies on the adoption of the system by websites to rate its sites, and the big problem with this is that currently not enough websites participate in it to really make this work. In 1999, the RSAC 'folded' into a new non-profit making organisation ICRA (Internet Content Rating Association) (URL-ICRA) with its primary aims of protecting children from harmful material, but preserving the nature of free speech on the Internet, and backing from many of the industry's leading names.

2.3.2 Monitoring

Monitoring software typically allows parents to monitor their child's online activity without necessarily limiting his or her access. Such software is often thought of as 'spy ware' since it often works invisibly in the background, without the users knowledge so that they continue in their activities as normal.

The benefit of monitoring software is that it allows a child to use the Internet to its full potential, without any of the problems involved with filtering which may incorrectly block sites that are in fact educational. Filtering can also only block access to chat rooms, or censor inappropriate words, without necessarily preventing a paedophile from soliciting private information about a child. As such, more freedom is given to a child to learn about and use the Internet in an responsible fashion, whilst providing parents with information which may be used to effectively monitor a child's online activity without them being there all the time.
This information may be highly valuable as it can be used to spot a child accessing inappropriate sites, or engaging in inappropriate conversation, which would otherwise not be spotted using either filter-only software or nothing at all. Information collated by monitoring software can then be used to discipline a child from accessing inappropriate sites on purpose, or prevent them from being targeted by a potential paedophile.

Most monitoring products incorporate some or all of the following functionality:

- Screenshot capture
- Internet activity monitor
- Key logging
- Record conversations in chat rooms and instant messaging
- Application monitoring
- Limit the amount of time spent online
- Operation in stealth mode

2.3.3 Current Problems

It has been reported several times in various articles that the current technological solutions involving filtering are inadequate since 'Underblocking' or 'Overblocking' will always exist (Willard, 2002):

1. Underblocking - No blocking technology is clever enough to block even 10% of the pornography on the Internet unless it effectively blocks most or all of the

materials on the Internet. This is because of the inherent complexity of human language and thought, not a matter of simply improving blocking technology.

2. Overblocking - Blocking technology always blocks more material than the small proportion of pornography it is able to block, thus significantly damaging the most basic and practical uses of the Internet, not to mention the free speech rights and civil liberties of every person accessing, publishing, or broadcasting on the Internet. Most of the material on the Internet is informative and useful and should not inadvertently or intentionally get blocked.

The implication of this is that no system will ever be 100% secure, and as such parents will have to make a choice (perhaps depending on the age of their child) on whether to have a system that underblocks, but gives more access to the vast quantities of information available, or to use a 'white list' filter system to 'overblock' but ensure that inappropriate sites are not accessed.

As filtering per se will not protect children from inappropriate conversation in chat rooms, unless access to all chat rooms are prohibited by the filter, an effective software solution must also incorporate functionality to monitor what a child's activities are on the Internet, alerting parents to behaviour that is deemed predatory or inappropriate. Analysis of current solutions in Chapter 3 indicate that although good solutions exist that manage either filtering or monitoring, there isn't a product available on the market that incorporates and applies both approaches effectively. This is the problem that this project will aim to solve, bearing in mind that a good solution must also be extremely usable to someone who is not computer literate.

2.4 Summary

This section begins by highlighting the significant strides made in Internet and computer technology in recent years, strides made so rapidly that we have in fact developed an 'information gap' where parents do not have the knowledge or the skills to supervise their children online. The section then moves onto looking at the potential dangers presented to children if left unsupervised and unmonitored when accessing the Internet. The physical dangers although remote, are very real, highlighted by some high-profile cases in recent times. Shocking statistics have also shown that the potential danger could be much higher as children remain uneducated about the dangers that may exist in meeting 'online' friends in the real world. The Internet also provides psychological dangers, through exposure to inappropriate material, which can also be as damaging to a child.

The section then moves onto the ethical side of this issue, highlighting legislation that has been introduced to deal with these problems, and also the various guidelines given by governmental institutions to parents and children. The morality of producing software to deal with these problems is also questioned and discussed in this subsection.

Finally, the section concludes with a high-level investigation into the current technological solutions to this problem, providing details of the main solutions of filtering and monitoring

The important issue that this section hopes to address is to clarify that this is a relatively new subject area in both the commercial and academic world, due to the rapid emergence of new technologies. This has hopefully been highlighted by references to rapid growth rates in Internet access, the rise in paedophilic activity on the Internet, new government guidelines and legislation and also the relatively new emergence of software that aims to deal with these

problems. To conclude, Dixon (2001) has stated that this area has definite investment potential in terms of research and development sponsorship into ways of improving the detection, traceability and removal of illegal material on the Internet.

2.4.1 Summary of Identified Problems

Chat Rooms

- Predatory behaviour from paedophiles
- Engagement in sexually explicit chat
- Use of foul and abusive language
- Disclosure of personal information
- Receiving of files which may be pornographic or malware
- Arrangement to meet strangers

Information disclosure

In chat rooms, email, websites, there is a problem of strangers soliciting 'sensitive' personal information including:

- Telephone / mobile phone number
- Personal email address
- Home / school address
- Surname
- Age
- Credit card details

Misuse of Computer

- Receiving files through email/chat
- Deliberately searching and viewing inappropriate or explicit material
- Playing games
- Downloading of illegal media files
- Using computer for 'hacking'

Spam / Email

- Receiving inappropriate images or links to URLs containing inappropriate material
- Problem of having multiple email service providers
- Similar problems as chat

Incomplete advice

- Informing children of SMART rules
- Informing supervisor of action to take
- Poor involvement of parents / authorities if required

Current Solutions

- Overblocking/Underblocking
- Lack of software which provides both filtering and monitoring effectively

3 Analysis of Current Solutions

The following section aims to describe and analyse some of the technical solutions available to parents to protect their children from the dangers of the Internet[13].

3.1 Filtering Software

3.1.1 MSN 8

URL: www.msn.co.uk (Accessed 7/12/03)

MSN 8 is a subscription service from that provides additional software that integrates seamlessly with existing Microsoft software. This software includes new parental controls over children when they surf the web or use e-mail, using age-based filtering. Parents can block or allow access to specific websites, allowing full control of how a child uses the Internet. Parents may also specify who can the child through MSN Messenger, a popular instant messaging tool, allowing pre-approval of anyone trying to contact the child, and detailing whom the child wants to talk to. The software also sends a weekly report that lists the total time spent online, all the websites that have been visited, and who your children sent MSN email or MSN Messenger instant messages to.

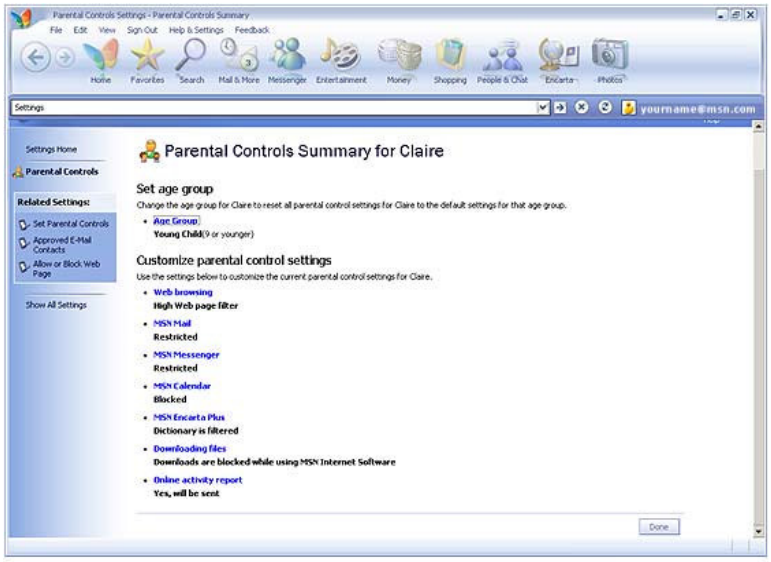

Figure 3.1 Parental control interface - MSN 8.

[13] With the exception of MSN 8 and Spector Pro 4 where trial software was unavailable.

One of the advantages of MSN 8 is that it is provided by Microsoft, implying that the software is typically easy for a parent to pick up due to the software having the same 'look and feel' of other Microsoft products. MSN 8 can also be configured so that if the child uses his/her MSN sign-in name from another computer, the controls that the parent set still apply. This is a big distinguishing feature that cannot be rivalled by comparable client-based software from rival companies.

The disadvantage of this service is that it is subscription-based, implying that there will be a continuous 'monthly' fee to pay, rather than a 'one-off' sum that many parents would prefer. The service also doesn't provide any protection to children who talk in chat rooms since doesn't provide parents with information about predatory behaviour, or keep logs of chat conversations for monitoring. It is also arguable that the features provided by Microsoft in this service should really be provided by default on the moral grounds that parents have already paid for Microsoft software, and as such parents should be provided with tools to protect their children online!

3.1.2 ActivatorDesk

URL: http://www.safesitesonly.com/ (Accessed 7/12/03)

ActivatorDesk is one of the few products on the market that offers 'White List' protection, that is, 'Safe-Sites-Only' only may be accessed, thereby giving 100% security. The product uses safe-lists created by non-profit organizations such as Dot Kids (URL-DK) who put child-safe websites into the '.kid' domain.

Analysing this product revealed that although it is easy to install, the actually usability of the administration tool is very low and would definitely confuse a user with little computer knowledge. The options screen that allows the administrator (parent) to define what should be allowed or blocked is confusing, incorporating many unnecessary features more geared towards desktop lock-down than Internet security.

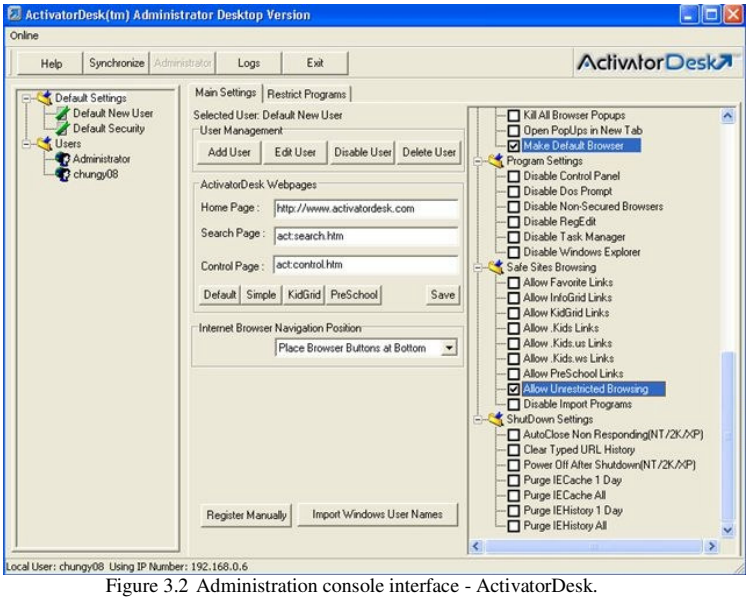

Figure 3.2 Administration console interface - ActivatorDesk.

Another negative is that the product only offers safe-list protection if the child accesses the Internet through the ActivatorDesk web browser (although Internet Explorer can be blocked from running by the software). This results in the child having to learn to use another web browser, giving no continuity or familiarity with software the child could use on another computer. Worse still, and more worryingly, after much difficulty configuring the browser to allow only 'kid-safe' websites, a search for 'sexes' on the ActivatorDesk browser revealed several web pages with objectionable material. This implies that either the browser is too difficult to set up correctly, or that the safe-list system is flawed. As a result of this, my overall impression is that this is a very poor product with virtually no 'positives' that can be taken from the evaluation and analysis.

3.1.3 Content Protect

URL: http://www.contentwatch.com/ (Accessed 7/12/03)

Ropelato (2003) rates Content Protect as the best family Internet filtering product of 2003 in his review that evaluates and compares 10 of the best products available in the market including products offered by the well known Net Nanny range and also software giants McAfee and Norton.

Content Protect is primarily a 'filtering' product that provides excellent functionality to filter and log web and chat access, allowing granular control over what should be allowed or blocked depending on various different categories, each of which can have a 'sensitivity' level defined. This built-in filter also works dynamically so for example, a particularly violent story in the news may lead to the BBC front page being blocked at one point during the day, but permitted at another point during the day. Content Protect also provides word recognition protection to foreign languages in its filtration system.

Figure 3.3 Administration console interface - Content Protect.

Product analysis pointed to the same conclusions drawn out by Ropelato, that this is in fact a very good filtering product. The user interface was very easy to use, allowing easy configuration even for the most illiterate of computer users due to its intuitive interface (not that much configuration is required out-of-the-box). Content Protect prevents any application from accessing the Internet before a username and password is given (this can be set to be blank for children who can simply click on their username). The username then allows appropriate web access and filter level depending on the predefined level given by the administrator (parent).

Figure 3.4 Username and password interface required
before web access is granted to an application – Content Protect.

Testing the filter proved to be quite enjoyable, as the product successfully blocked virtually everything during the test, even at search engine level. For words which did slip through at search engine level, clicking on an inappropriate website resulted in a warning message stating that the site had been blocked. Anything that is blocked is then logged, allowing the parent to view in the 'Report' section what sites had been denied, with the opportunity of viewing it themselves and adding it to the 'Allowed' list if it is deemed suitable. Finally, Content Protect also provides remote management, allowing the settings and logs to be changed or viewed respectfully, from a remote machine. The reports incidentally are given in web-format using graphical displays, whilst logs are given as a list of events.

One of my disagreements with Ropelato is over his view that the remote management functionality is extremely impressive. Although it does allow settings and logs to be changed or viewed remotely, say from an office or whilst on holiday, it lacks 'real time' monitoring functionality for a parent to actively check the content of online conversation in chat rooms. In connection with this, although it logs whether the child has been chatting in a chat room, it provides no filtration against inappropriate conversation and logs are not kept of these conversations so a parent would have no knowledge if anything predatory or inappropriate were said. However, despite my reservations over these last 2 issues, this product does offer superb filtering and access control functionality to the Internet.

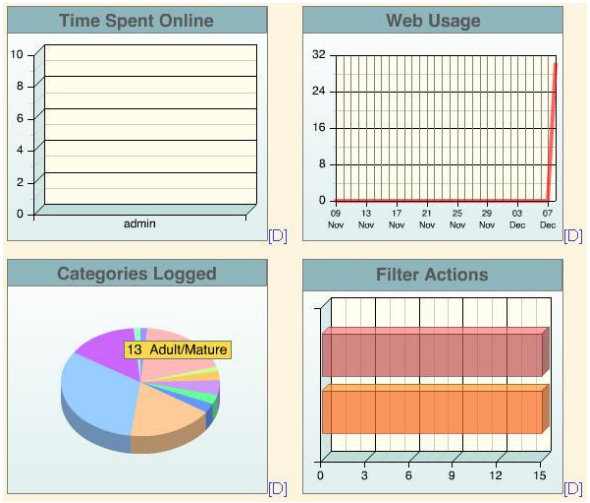

Figure 3.5 Details of web activity – Content Protect.

3.1.4 Cybersitter 2003

URL: http://www.cybersitter.com/cybinfo.htm (Accessed 8/12/03)

Munro (2003) rates Cybersitter 2003 as PC Magazine's Editors' Choice for protecting children online. Like Content Protect, Cybersitter is primarily a filtering product which provides an extensive range of pre-defined categories from sex and drugs to cults and hacking, making fine-tuning what may be accessed by children simple.

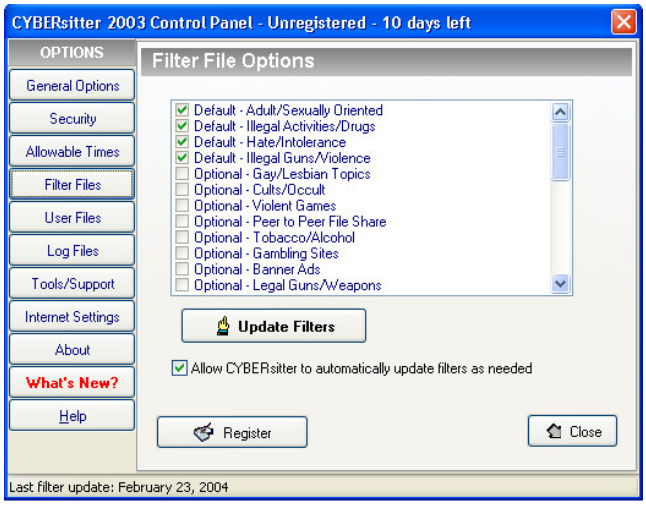

Figure 3.6 Web filter options – Cybersitter.

Further functionality on top of what is provided by Content Protect is the ability to add custom phrases and sites to the predefined list that Cybersitter then proceeds to filter on top of the predefined categories. Such functionality is very useful when the majority of sites in a certain category are deemed suitable, but the supervisor wishes to prevent access to particular sites within that category. This functionality also extends to chat in that Cybersitter will censor bad words or phrases being received or sent by the child. Conversely, functionality is

also provided to allow access to custom defined ‘Allowable Sites’ which allow access to sites which would otherwise be blocked within a category.

Figure 3.7 Custom Bad sites, words, phrases blocker - Cybersitter

Product analysis revealed that Cybersitter was not as impressive as Munro had made it out to be, and the author was left with the impression that Cybersitter had a very ‘amateurish’ feel about it. To begin with, the administration console looked more shareware than what is expected given the $39.95 price. Although Cybersitter does implement a website filter, unlike Content Protect which prevents objectionable sites from being listed at search engine level, Cybersitter continues to display ‘found’ sites from the search and only blocks it after a link has been clicked. When tested on an inappropriate site, it simply failed to load it, giving a default Internet Explorer ‘Page Connot Be Displayed’ message rather than informing the user explicitly that the site had been barred from access.

Additionally, although the ability to log chat is listed, this functionality is severely limited to simply displaying the time, application, keywords detected and user. The actual conversation is not recorded, implying that the log data is effectively useless in determining whether anything inappropriate has taken place, and also useless as means of disciplining a child using hard evidence.

A positive aspect of Cybersitter is its ability to work stealthily in the background after configuration. Cybersitter can be configured to either sit in the system tray or act completely transparently as a service in the background. This implies that unlike Content Protect, a child is not required to enter a username and password in order to access the Internet. Another positive is Cybersitter’s ‘block key words and phrases’ functionality, which strips out these key words and phrases from email and chat leaving a blank in place of the offending key. This functionality is useful to protect a child from conversing using inappropriate language and could potentially be extended to prevent the transmission of sensitive information.

3.1.5 Net Nanny 5.0

URL: http://www.net-nanny-software.com/index.asp (Accessed 8/12/03)

Net Nanny is one of the leading and most established brands in the online child protection market. From the list of applications tested, Net Nanny had by far the most impressive range of features ranging from website blocking, and chat logging to configuring access times and user configurations. Blocking level is defined using a slider that allows an administrator to specify the level of blocking to apply, from least restrictive (which blocks nothing) to most restrictive (which results in access only to websites defined as 'Allowed').

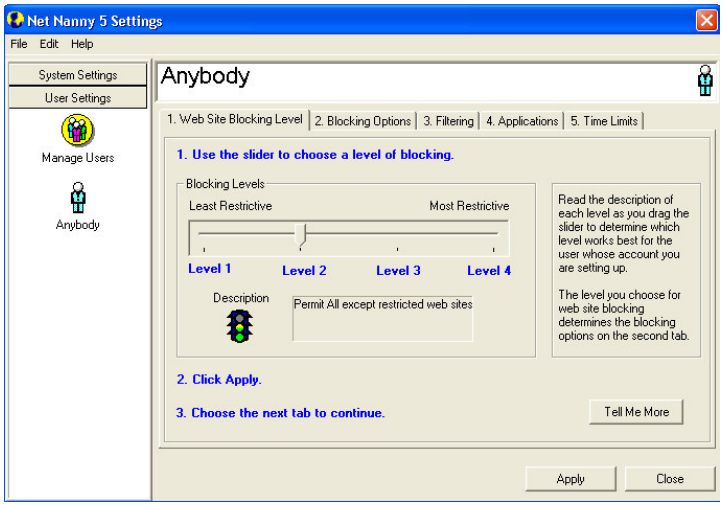

Figure 3.8 Website blocking level – Net Nanny

A unique feature of Net Nanny is its functionality that allows an administrator to enter personal details about a user such as home address, telephone number and credit card details. Net Nanny will then prohibit this information from being transmitted in email or chat, thus providing a further level of security over what is offered by its main competitors.

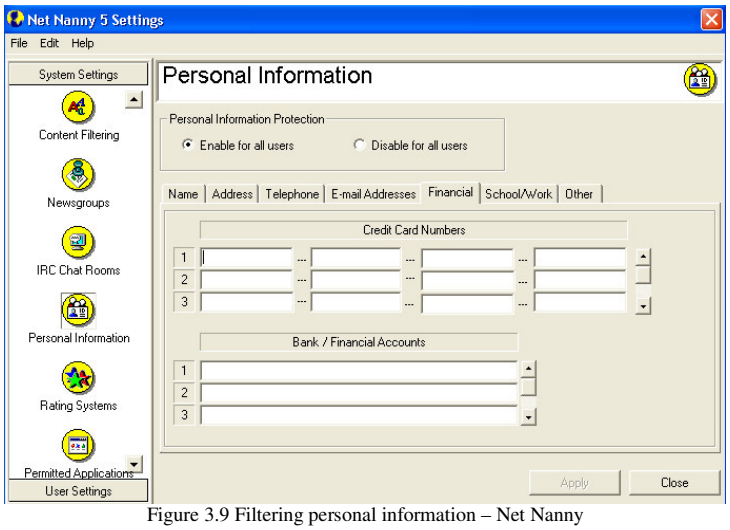

Figure 3.9 Filtering personal information – Net Nanny

By default, Net Nanny provides a baseline 'Anybody' configuration where the default configuration applies. Different levels of protection for up to 12 children can then be defined, each with their own configuration settings giving a high level of configurability. Testing showed that although the functionality was provided, its configuration was both confusing and inefficient compared to professional user management software such as Windows Server. Improvements could include the use of 'groups' where configuration is applied to a group and users residing within a group get the group policy. As with Content Protect, each user also has to specify his/her own password in order to access the Internet. In order to make protection more transparent, a users configuration should be tied onto his/her logon username.

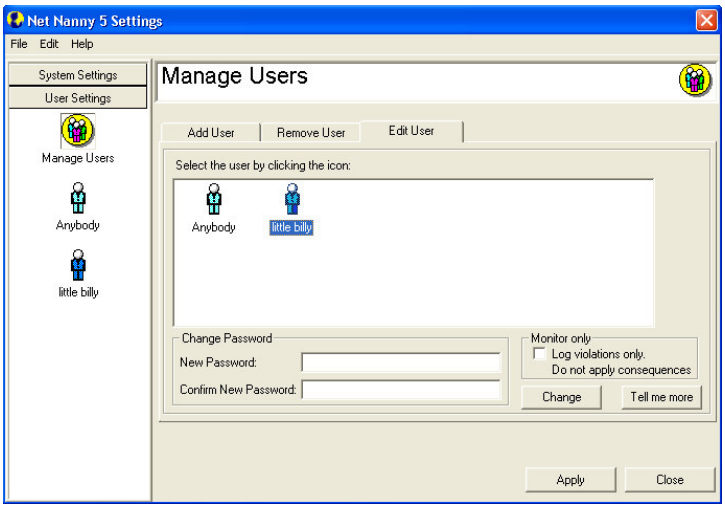

Figure 3.10 Managing users – Net Nanny

Ropelato (2003) lists Net Nanny's installation and configuration as one of the products weaknesses, and having tested the product, the author is inclined to agree with Ropelato's findings. Given the number of options on offer, configuring Net Nanny can be confusing, especially to a novice parent with little computer know-how. On some of the configuration screens, Net Nanny has simply crammed 'too much' functionality onto a single page and as such, the pages look both cluttered and difficult to follow. Also among its weaknesses are its poor logging facilities, and as with Content Protect and Cybersitter, its inability to log conversations flagged with inappropriate material. It is noted however, that Net Nanny does offer a separate product 'Chat Monitor' which is reported to offer such functionality.

3.2 Monitoring Software

3.2.1 Spector Pro 4

URL: http://www.spectorsoft.com (Accessed 8/12/03)

Spectro Pro is rated by Glass (2002) of PC Magazine as the best Internet monitoring solution on the market at the time of review. As a 'descendent' of Trojan horse program Netbus (a popular tool used by hackers to gain remote control over a computer system), Spectro Pro is described to be feature-rich in monitoring functionality such as logging key strokes, capturing

what websites are visited (including screenshots), and, monitor chat and instant-messaging conversations. It is also described to be the 'stealthiest' of products reviewed, implying that it is well hidden from the user so that monitoring activity isn't suspected.

Unfortunately, first-hand analysis of this product wasn't possible, although examining the information provided by the manufacturers SpectorSoft reveals that Spectro Pro would provide excellent monitoring functionality not provided by any of the other products analysed. Of particular importance is Spectro Pro's ability to record conversations in chat rooms and instant messaging providing parents with a clear record of conversations which have taken place, allowing identification of anything inappropriate (including the username of whom the conversation took place with).

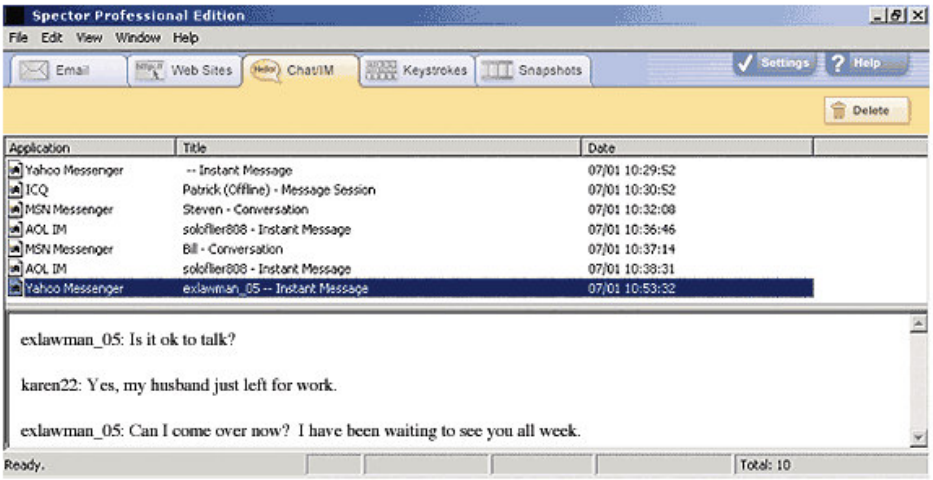

Figure 3.11 Conversation logs - Spectro Pro 4

Finally, one of the key features that really distinguishes this product, is the ability to create a 'notification list' which monitors chat, web browsing and email for key words or phrases and alerts the parent whenever the word or phrase is detected. This allows the parent to be notified by email when an occurrence of keyword detection is detected. Spectro Pro also logs all activity around that time, maximizing the amount of information provided so that the full context of the information is understood. This functionality is extremely valuable due to its ability to significantly cut down the amount of time that a parent would otherwise spend manually analysing the logs and information captured.

As first-hand analysis of this product wasn't possible, it is difficult to assess how effective this product is in reality, but looking at the features it provides, it definitely offers many benefits to parents wishing to keep a supervisory eye over their child's online activities. One of the weaknesses of this product however, is that it does not provide any content filtering functionality – since it works stealthily in the background. As such, many inappropriate websites would be accessible to a child browsing the Internet, unless a separate filtering product is used adjacently to Spectro Pro.

3.2.2 SnapKey

URL: http://www.spy-gadgets.com/parental-control/snap-key/snapkey.htm
(Accessed 8/12/03)

SnapKey was examined as 'monitoring' software due to its promise in being able to log web access, chat conversations and capture screenshots. Marketed as 'undetectable', SnapKey uses an interface that is stored on a floppy disk, as apposed to the client system in order to minimise detection.

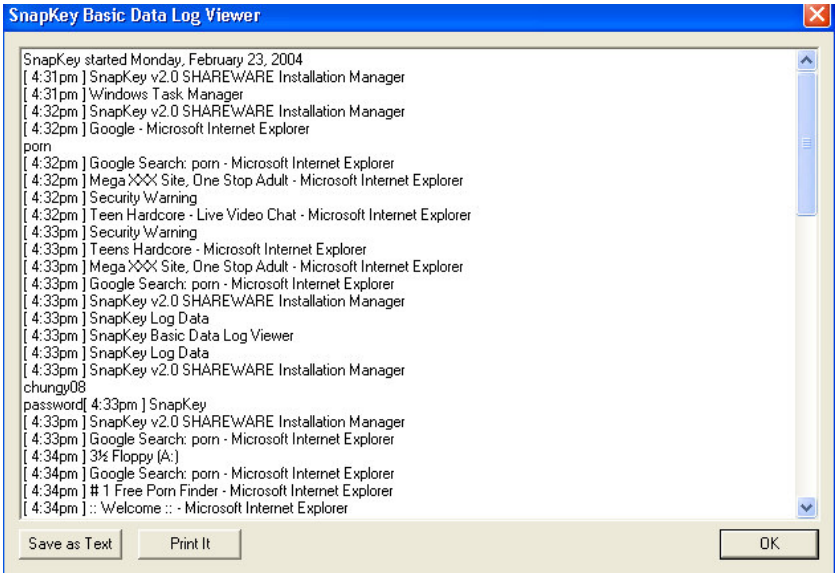

Figure 3.12 Log transcript - SnapKey

During testing, this product proved to be very poor, in both functionality and the user interface provided. No filtering of any form is provided, with the essence of the product purely providing monitoring of web access and chat transcripts only. Given that logging is the only thing this product does, one would expect the logs to be richer in terms of information presented.

3.3 Summary

From the analysis conducted, it is apparent that there is currently a gap in the market for a product that effectively combines filtering and monitoring. Most of the filtering products examined appeared to be very strong in their ability to restrict the content that may be accessed, but were far less effective in their capabilities as chat conversation loggers. Conversely, monitoring software that specialised in providing chat-logging functionality did not contain filtering functionality to prevent access to inappropriate websites.

As such it is proposed that the application to be developed should incorporate functionality that allows both filtering and monitoring on a client system.

In addition, it should also be mentioned that remote access functionality would be extremely valuable - something the vast majority of products do not provide. Understandably, the current requirement for remote access is relatively low, but given the decrease of computer hardware prices, the increasing awareness of home networking and the sharing of Internet connections, such functionality can become very useful in the near future. It can also be mentioned that this functionality would also be extremely suited for school environments where a 'supervisor' would typically look after a big group of children accessing the Internet. Remote access can be defined to consist of two parts: remote configuration, which allows a central server to remotely configure settings of a client desktop; and remote desktop control, which allows an administrator to view a client desktop remotely and take control if necessary.

Accordingly, it is also proposed that remote access functionality be included in the requirements section of this document, although the constraining factors of technical expertise and time will mean that remote access functionality implementation will be limited to remote configuration of client desktops only.

3.3.1 Summary of Identified Strengths

- Ability to allow and deny web access depending on pre-defined categories of websites
- Ability to allow custom definition of allowed and blocked websites which expand on pre-defined lists
- Ability to specify additional bad words and phrases to block from chat and email
- Ability to censor personal data from being transmitted in chat and email
- Ability to log entire chat transcripts based on detection of keywords
- Ability to log web access violations
- Ability to define different configurations for different users

3.3.2 Summary of Identified Problems

- Absence of solutions offering both filtering and monitoring
- Absence of remote configuration functionality
- Absence of remote access functionality
- Poor user group management and rule configuration
- Poor integration of existing solutions and underlying operating system, requiring secondary logon to access Internet
- Logs not information rich
- Cases of poor unintuitive user interface
- Cases of overcrowded configuration screens
- Requirement to use custom web browser

4 Requirements

4.1 Requirements Analysis

Stiller (2002) defines requirements analysis as critical to producing successful cost-effective software, helping to validate and verify that the software is correct. Given the importance of requirements analysis, several techniques were used, enabling the construction of an unambiguous specification that will provide a solid platform on which to design and build the proposed system:

1. The conduction of the literature review, to fully understand the application domain and the problems posed.

2. The examination of functionality of existing systems, to derive the strengths and drawbacks of current solutions.

3. The development and analysis of use cases derived from the first two techniques to determine the functional behaviour of the system.

The procedure of performing the first two analysis techniques have enabled the development of several high-level requirements:

- Ability to log email and chat conversations
- Ability to define rulesets to specify:
    - Blocked/permitted URLs
    - What should be logged (chat, email, URLs, downloads)
    - What keywords are deemed of an abusive nature
    - What keywords are deemed to be sexually orientated
    - Severity of prohibited activity
    - Alert type and priority
    - Sensitive information list
- Ability to alert the supervisor if a rule is broken
- Prevent sensitive information from being transmitted
- Inform child (user) if a URL has been blocked from access
- Display logged information in a meaningful way to the supervisor
- Client system will upload logs to a central server
- Client system will download most current ruleset on start-up
- Client system should record information about ‘predator’ such as email address
- Client system must operate transparently in the background
- Supervisor should be able to grant access to URL on request
- Users should be prevented from tampering with ruleset configurations
- Supervisors shall require a username and password combination to access the main console
- Supervisors should be able to access and control a remote computer

These high-level functional requirements can be further analysed through use case analysis to provide a more detailed definition of required system behaviour. Sommerville (1993) defines use case analysis as a scenario-based technique for requirements elicitation to represent possible interactions that will be defined later in the system requirements specification. Sequence diagrams may also be used to add information to a use case, showing how the actors involved interact with the system, and the operations associated with these objects. Appendix A contains a list of key use case scenario's, together with their respective sequence diagrams.

4.1.1 Functionality-Prioritization Analysis

Given the wide scope of possible functionality, together with the time and resource constraints of the project, it is necessary to perform an analysis of the most important and critical requirements that must be achieved in order for the product to be deemed successful. Initial assessment of high-level requirements revealed that it would be infeasible to develop an application covering all of email, chat and web access. It was decided to drop email functionality from further consideration at this stage, given that its very nature allows records of messages received and sent to be archived in the inbox and sent-items folder respectively.

Additionally, given the immense range of chat services, it would infeasible to implement a solution that covered every service. It was decided that for the purpose of designing a prototype solution, MSN Messenger[14] would be studied and analysed. The justifications for this decision are as follows:

1. The vast majority of the author's contacts are on this service, allowing their participation during service analysis and testing to simulate realistic chat conversations.

2. The nature of MSN Messenger chat results in easier analysis of chat packets than that of a group-chat services which typically involves up to 50 participants. The consequences of this are that the analysis of such conversations would involve filtering through hundreds, rather than thousands of packets to identify and determine the packet structure and procedure of conversations.

3. Since the proposed solution would be a prototype demonstrating proof-of-concept, a successful implementation of monitoring functionality for one chat service protocol would provide a framewrok for the future addition of supplement chat services.

Further prioritisation of requirements resulted in the insertion of use cases developed in section 4.1 into a matrix to elicit the most important functions that must be implemented. Less important functionality will not be implemented in the final application due to the aforementioned constraints. A prioritisation matrix (Table 4.1) shows options narrowed down through a systematic approach of comparing between options measured one against another, and assigning a weighting for each option.

[14] Available at: http://messenger.msn.com/ (Accessed 7/4/04)

The results of the analysis indicate (in order of importance) the following:

1. = Block access to inappropriate URLs
   = Block inappropriate chat activity
3. = Remote definition and application of ruleset configuration
   = Remote view of violation logs
5. Password protection of administration console
6. = User group configuration
   = Permit access to URLs on request
8. Allow administrator remote desktop accesses

The results indicate that the most important system requirements were the development of blocking mechanisms for website access and chat conversation on the client system, followed by the requirement to configure and view violation logs from a remote server. It was also decided that password protection for the administration console would also be implemented to provide extra security to the application. The implementation of user group configuration, URL access on request and remote desktop functionality were deemed the least important and as such will be overlooked during the implementation phase due to project constraints.

	Admin Console Logon	Ruleset Configuration	View Violation Logs	User Group Configuration	Remote Desktop	Permit Access on Request	Block URL Access	Block Inappropriate Chat	**Total Weighting**
Admin Console Logon		0	0	1	2	2	0	0	**5**
Ruleset Configuration	2		1	2	2	2	0	0	**9**
View Violation Logs	2	1		2	2	2	0	0	**9**
User Group Configuration	1	0	0		1	1	0	0	**3**
Remote Desktop	0	0	0	1		1	0	0	**2**
Permit Access on Request	1	0	0	1	1		0	0	**3**
Block URL Access	2	2	2	2	2	2		1	**13**
Block Inappropriate Chat	2	2	2	2	2	2	1		**13**

Table 4.2 Functionality-Prioritisation Matrix

Weighting:

- If the functionality is more important that the other, it has a weighting of 2
- If the functionality are of equal importance, assign each a weighting of 1
- If the functionality is less important than the other, it has a weighting of 0

4.2 Requirements Specification

Stiller (2002) defines requirements specification as a formal expression of needs, setting out the system services and constraints in detail. Following the process of requirements elicitation and analysis, this section aims to express the key functional needs of the proposed application in detail.

4.2.1 Definitions

- System

  It is apparent from requirements analysis that the proposed software component will consist of two parts:

  a) SecureNet - the client-side component responsible for 'doing the work' that is blocking websites, recording chat conversations and uploading logs to the administration console (SecureNet Admin) when violations are detected.

  b) SecureNetAdmin - the server-side component responsible for the configuration of rules and its distribution, the display of log information, and general administration of SecureNet.

- Administrator

  The user performing the supervisory or parental role. This role involves the configuration of rules using SecureNet Admin, and is responsible for examining violation logs.

- User

  The end-users of SecureNet. The target-audience of SecureNet is children of any age who use the Internet to browse websites and use chat applications.

- RuleSet

  The defined configuration of SecureNet. The configuration ruleset will include: lists or permitted and prohibited websites, prohibited words and phrases, sensitive information, logging options.

- Logs

  A record of net and chat activity.

- Chat

  The process of conversing online using chat programs or chat rooms.

4.2.2 SecureNet System Features

4.2.2.1 Filter URL access

The first of two primary features, SecureNet must be able to filter the websites accessed and compare its URL with those held in a pre-defined list. The pre-defined list will consist of:

- Pre-defined websites based on Categories including: Adult/Mature, Drugs, Entertainment, Email, Financial, Gambling, Games, Kids, Music, News, Politics, and Science. The administrator will be able to define these categories as 'Allowed' or 'Blocked'.

- Custom-defined websites, which are defined in a 'Custom Allowed List' or 'Custom Blocked List'.

SecureNet will allow filtering on 4 levels:

1. High. The most restrictive security level for web browsing, allowing access to web sites defined on the administrator's 'Custom Allowed List' only. This level is appropriate when web use is to be restricted to selected sites only and appropriate for very young children.

2. Medium. A high security level for web browsing, allowing access to web sites pre-approved by SecureNet and by the administrator. This level is appropriate when web use is to be restricted to known safe sites and appropriate for young children.

3. Low. A safe security level for web browsing, allowing access to web sites which are not explicitly blocked by SecureNet and the administrator. This level is appropriate for restricting access to inappropriate web sites and appropriate for older children.

4. Off. The most relaxed security level for web browsing, allowing access to all web sites without restriction. Appropriate for adults.

If access to a website is denied, a 'Access Forbidden' page will be displayed to the user informing him/her that the desired page is prohibited and further attempts to access it will result in disciplinary action. This requirement ensures that access to inappropriate and objectionable material is denied to the child.

4.2.2.2 Monitor Chat Conversations

The second of the primary features, SecureNet must be able to monitor MSN Messenger chat conversations and compare content with words and phrases (keys) in a pre-defined list. Conversations will be recorded on the commencement of a conversation until its cessation. If no keys are detected, the record of the conversation is deleted unless otherwise specified by the logging option, otherwise, if keys are detected, the conversation is saved. If keys are detected in the conversation, SecureNet must provide functionality to censor inappropriate content by replacing the detected key, or the replacement of the entire message. These

requirements ensure that chat of an inappropriate nature is logged, prevents the participation in abusive or sexually oriented chat, and prevents the release of sensitive information.

4.2.2.3 Logging

SecureNet must have the ability to store logs locally and upload these logs to the central server (SecureNet Admin) for administrator inspection. Web and chat logs will be stored separately as they contain different types of information. An administrator must be able to define the criteria for logging to occur, for example, an administrator must be able to define if logs are to be taken verbosely, that is to log all websites visited or chat conversations participated in; or to log web access when an attempt is made to access a prohibited site or when keys are detected in a chat conversation. Additionally, it must also be possible for an administrator to turn off logging altogether. An administrator should also be able to define the frequency at which logs should be uploaded to the SecureNet Admin server.

4.2.2.4 Transparency

An important non-functional requirement required of SecureNet is the ability to operate transparently. The application must not obtrude in normal system operation and should not be noticeable to the user. This requirement ensures that the user may continue Internet use in a normal manner without the hindrance of the SecureNet application on the desktop or taskbar.

4.2.2.5 Information

Finally, in compliance with government guidelines on user information, it is required that the user should be warned that monitoring is in process on the system and unacceptable Internet activities should be discouraged. In addition, following the HCI principle of ‘Synthesizability’ as defined by Dix (1993), the user should also be informed if an attempted access to a website is blocked, thereby giving the user a true perception of system state, rather than relying on a generic web browser error message.

4.2.3 SecureNet Admin System Features

4.2.3.1 Password Protection

Access to the console should be granted on the provision of a correct username and password pair. This requirement prevents unauthorised persons from accessing SecureNet Admin, and viewing logs of website activity and chat transcripts, thus keeping details confidential. Administration credentials also prevent unauthorised change of ruleset configuration. Additionally, it must also be possible to create, edit and delete administrator details from the SecureNet Admin console.

4.2.3.2 Logging

It is necessary that the SecureNet Admin console should allow the continuous receival of logs. As such, it is a requirement that SecureNet Admin should be able to work transparently in the background. The administrator should be notified immediately on the receipt of new logs. This requirement ensures that violation log data is recent and relevant, allowing action to be taken quickly if needed.

A primary function required of SecureNet Admin is the display of log information. In order to allow sorting and filtering functionality, log data should be displayed in a tabular format, allowing the user to click on a column in order to sort that particular column. When viewing chat log data, chat transcripts should be displayed in a separate window so that the user can easily view the entire transcript. Functionality must also be included to allow the deletion of log data from the system. This requirement addresses the need for a comprehensible format of displaying log data to the administrator.

Finally, it must be possible for an administrator to extract log information from the SecureNet Admin system to a text file that may be printed, or distributed via email to a parent.

4.2.3.3 Ruleset Configuration

Another primary function required of SecureNet Admin is the ability to define the configuration of SecureNet. Configuration options must include at least:

- SecureNet Admin server IP address, informing SecureNet clients' of the location from which to retrieve configuration updates and the location to send violation logs.
- A list of pre-defined categories containing websites which fall into certain genres such as sex, violence, political, educational etc. The administrator is then permitted to define whether access to a specific category is to permitted or denied.
- Custom blocked and allowed lists, allowing the definition of additional websites to be blocked or allowed on top of the pre-defined lists.
- Prohibited words and phrases list, allowing the definition of words and phrases that are deemed unsuitable for chat conversations.
- Sensitive data list, allowing the definition of personal information that must be prohibited from publication during a chat conversation.
- Log options, allowing the specification of logging criteria, and frequency of log uploads for each of web and chat configuration.

It must be possible for an administrator to change existing configurations by adding new entries, or by editing or deleting existing entries in custom lists. As a corollary, it must be possible for the administrator to enforce new ruleset configuration immediately by pushing rules out to SecureNet clients. This requirement ensures that each SecureNet client has the most up-to-date ruleset configuration definition.

4.2.3.4 Appearance & Layout

An important non-functional requirement of the SecureNet Admin console is to ensure that the system is as learnable and usable as possible, given the technical expertise of the target users. As such, it is important that its appearance and layout follow general usability guidelines including familiarity and consistency as defined by Dix (1993).

To comply with familiarity guidelines, the user interface of SecureNet Admin should be designed with a similar look and feel style of existing Windows software. This envelops all of colour schemes, icons, button behaviour, and screen layout presentation.

To comply with consistency guidelines, it follows that the design of screen layout and error messages are similar between screens. Additionally, the use of colour within the application should also remain consistent.

4.2.4 Constraints

4.2.4.1 Operating Environment

The application will be implemented and tested using two computers running the Windows XP operating system with version 1.1.4322 of the .Net framework. Both computers have a minimum hardware specification of: Pentium 3 class processor running at 500Mhz with 256MB Ram and a plethora of hard disk space. Additionally, Internet access during development will be through broadband Internet access using Ethernet networking. Guarantees cannot be made that the application will operate on dissimilar systems.

4.2.4.2 Programming Language

The application will be developed using C# and the .Net framework. Choice of implementation language was heavily influenced by the need for strong network functionality and rapid graphical user interface creation capability. Both Java and C# were identified as suitable choices, but C# was chosen on the basis that its association with the .Net framework provided integration with the Windows platform, giving enhanced look and feel possibilities in graphical user interface creation through the use of Windows Forms, and easier implementation of Windows networking technology through the use of C# socket programming. Additionally, C# applications have faster execution speed and have the ability to run ‘as a service’ allowing the satisficing of the application transparency requirement.

4.2.4.3 User Characteristics

The user in this context is that of the administrator. It is assumed that the administrator will not be an expert in computer use or application configuration, implying that the application must be easy and intuitive to use. The involvement and participation of the user during application development is considered beyond the scope of the project, as is the production of user documentation.

5 Design

5.1 System Overview

From the integration of use case diagrams in Appendix A, it is possible to construct a high-level overview of system design. Referring back to section 4.2.1, the proposed system will consist of two components SecureNet (the client-side filtration system) and SecureNetAdmin (the command point to configure and distribute rulesets and view collated log data).

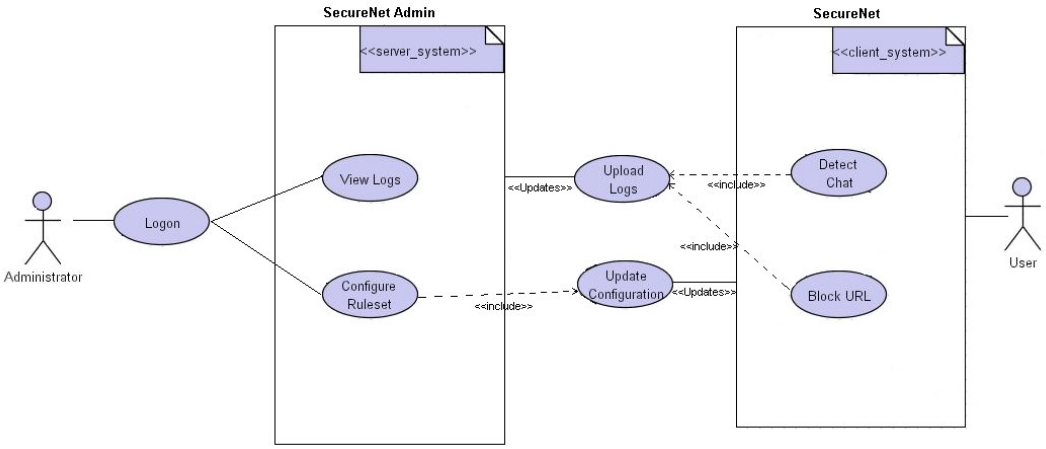

Figure 5.1 System Overview

Taking each component as a starting block of class design, each use case will be examined in additional detail allowing definition of the underlying design, the interactions involved, and where applicable, the user interface design.

5.2 SecureNet Class Design

5.2.1 SecureNet System Overview

It is obvious that a packet analyser will be a core component of the SecureNet system. The packet analyser will be responsible for filtering web access request's, and also responsible for filtering chat conversations. The filter rules that will be applied to the packet analyser will be defined by the SecureNet Admin system, and as such, SecureNet must have a mechanism that allows the retrieval of configuration updates. Additionally, SecureNet must have a mechanism with which to upload logs to the SecureNet Admin system.

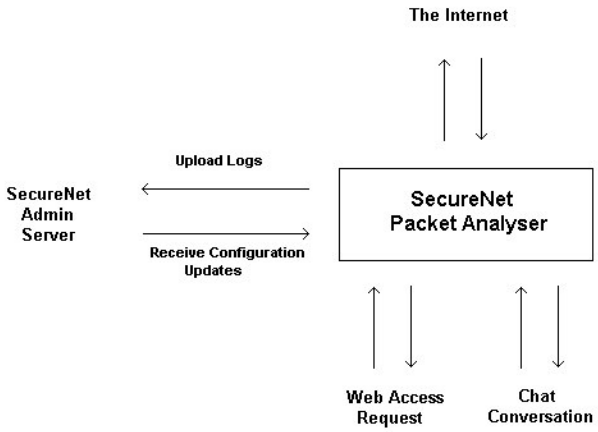

Figure 5.2 SecureNet System Overview

5.2.2 SecureNet Packet Analyser

The packet analyser is the core component responsible for filtering Http web access queries and monitoring chat conversations. In order to allow web access filtering and chat monitoring, the design of the analyser must first incorporate functionality to load ruleset configuration data including IP lists, prohibited words, phrases and sensitive data. Additionally, the analyser must load the configuration of log options, which defines what type of logging must take place in the event of a violation.

In order to implement filtering and monitoring functionality, we must first understand the structure of the relevant packets. The structure of Http packets is trivial, and Forouzan (2003) defines a Http request message to consist of a request line, a header and sometimes a body:

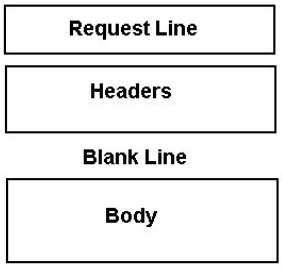

Figure 5.3 Http Packet Structure

We are predominantly interested in the request line, as it contains details of the URL the user is attempting to access:

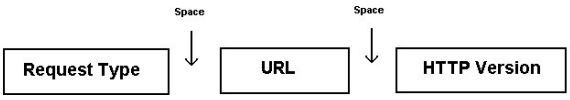

Figure 5.4 Request Line Structure

By parsing the request line, we can extract the URL and use DNS to resolve its IP address. Depending on the security level at which SecureNet is set, we can then do the following comparisons with the pre-defined IP lists:

1. If the security level is set to 'High':

    - If the IP address matches a custom-defined 'Allowed' address, allow the request through the packet analyser, otherwise drop the request and return a 'Blocked URL' page to the user.

2. If the security level is set to 'Medium':

    - If the IP address matches an address defined as 'Allowed' in the SecureNet category list, or an address which is custom-defined as 'Allowed', allow the request through the packet analyser, otherwise drop the request and return a 'Blocked URL' page to the user.

3. If the security level is set to 'Low':

    - If this IP address matches an 'Allowed' IP address, allow the request to go through the packet analyser.
    - If the IP address matches a 'Blocked' IP address, drop the request and return a 'Blocked URL' page to the user.
    - Otherwise, if the IP address does not match in either list, allow the request to go through the packet analyser.

4. If the security level is set to 'Off':

    - Allow the request to go through the packet analyser.

The type of logging to conduct varies depending on whether the option is set to 'All' (which logs all requests), 'Log Blocked' (which logs access to blocked websites), or 'Off' (which logs nothing). If logging is to take place, the information that must be logged includes:

- User – the name of the user
- URL – the name of the website which the user attempted to access
- IP – the IP address of the website which the user attempted to access
- Action – indicating whether access was permitted or not
- Time – the time of attempted access

To decipher the packet structure of MSN Messenger conversation protocol was more difficult, requiring the utilisation of Ethereal [15], a freely available packet analyser. A detailed analysis

[15] Available at: http://www.ethereal.com/ (Accessed 22/4/04)

of the MSN Messenger Protocol from packet logs is available in Appendix B. The structure of MSN chat packets is summarised as follows:

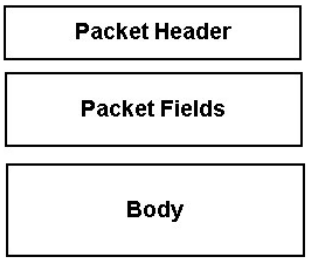

Figure 5.5 MSN Messenger Packet Structure

5.2.2.1 Packet Header

A MSN packet header begins with a command, consisting of the following types:

- CAL - indicates that the local user is attempting to contact someone on their contact list
- RNG – indicates that the Messenger service is attempting to establish a chat connection
- ANS – indicates that the connection has been accepted
- JOI – indicates a local user joining a chat conversation
- IRO – indicates a contact user joining a chat conversation
- MSG – indicates a packet with conversation details
- OUT – indicates the user has left the conversation

(There are other existing commands but their discussion is beyond the scope of the project)

We are particularly interested in packets with the 'JOI' and 'IRO' command, as they indicate the commencement of a conversation, thereby informing us when to start the logging process, and also the 'OUT' command, which indicates the termination of a conversation, thereby informing us whether to keep or dispose of the log depending on whether a flag was raised due to the detection of inappropriate conversation. Additionally, we are interested in packets with the 'MSG' command as they signify chat conversation content.

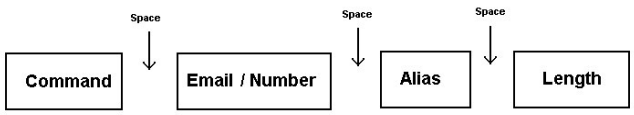

Figure 5.6 MSN Messenger Packet Header Structure

For packets containing conversation content, the command type will be of type 'MSG', and following this the packet header will contain:

- Email / Number. If the packet has come from the contact, this part of the header will contain the contact's email address, otherwise if the packet has been sent by the local user, this part of the header contains a number indicating the sequence of the packet.

- Alias. If the packet has come from the contact, this part of the header contains the contact's online chat alias. If the packet has been sent by the local user, this part of the header contains a 'U' to indicate if the local user is typing, or a 'N' to indicate if a message has been sent.

- Length – the length in characters of the current packet.

By parsing the packet header, we can extract information about the contact's email address and alias, allowing a reference to a logged chat conversation.

5.2.2.2 Packet Fields

The packet fields allow the transmission of additional information, including:

- MIME-Version – indicating the MIME version used in the conversation

- Content-Type – can be either:

    - text/msmsgscontrol – indicating that the user is typing
    - text/plain; charset=UTF-8 – indicating the character set of a transmitted message

- This field can be one of the following:

    - TypingUser – indicating the user that is typing a message
    - X-MMS-IM-Format – indicating the format of the conversation

We can use this information to decide whether the packet contains actual chat conversation content, or merely indicates that a user is typing. Additionally, the packet fields' section terminates with a double newline sequence, denoted as '0d 0a 0d 0a' in hexadecimal, or '\r\n \r\n' in standard type. As such, we can design the analyser to recognise this sequence in any packet containing the 'MSG' command and inspect whether or not chat conversation content is present in the packet body.

5.2.2.3 Body

If the packet contains a chat conversation, the content of this conversation goes in the body after the double newline sequence, otherwise if the packet is of type 'TypingUser' the body is empty, and we can ignore the packet from the log. It is now possible to define an algorithm (to be discussed in detail in Section 6: Implementation) to parse each conversation packet into its constituent fields and extract the conversation message from the body into a format we can keep in a log.

5.2.2.4 Detecting Violations In Chat Conversations

As we are now familiar with the structure of MSN chat packets, and have a high-level design method for extracting the conversation content, the next step in the process is to configure the analyser to compare this content with a list of inappropriate words and phrases which have been defined as abusive or sexual in nature, and also with a list of sensitive information. If inappropriate content has been detected, a generic 'found' flag should be raised. This flag will therefore provide the functionality to only log conversations with inappropriate content.

The type of logging to conduct again varies depending on whether the option is set to 'All' (which logs all conversations), 'Log Conversations containing Inappropriate Material' (which logs conversations where the 'found' flag is raised), or 'Off' (which logs nothing). If logging is to take place, the information that must be logged includes:

- User – the name of the user
- Contact – the email of the contact who the user is talking to
- Violation Type – indicates whether inappropriate language was used, or if sensitive information was released
- Keys Detected – the prohibited keys which were detected in the conversation
- Time – the time of offence
- Message – a transcript of the entire conversation

5.2.2.5 Censoring a Message

Another function required of the chat monitor is the ability to censor the keyword or replace the entire message with a generic censor message, should inappropriate content be detected. An extension of the previous function, if inappropriate key words are detected in a conversation message, depending on what censor option is selected, we can replace the entire message with the defined censor message, or alternatively go through the message detecting keys, and replacing each key with a predefined censor string. Finally, we must rebuild the packet using the defined MSN packet structure, and inserting the censored conversation message in place of the original message, before forwarding it onto the user.

5.2.3 Configuration Listener

A subsidiary function of SecureNet is the ability to receive new, updated ruleset configurations from the SecureNet Admin server. New rulesets may include information concerning new IP addresses to allow or block, new inappropriate words and phrases, new sensitive information, and also logging options. As such, it is important that SecureNet has the ability to remain up-to-date with the latest configuration.

By default, SecureNet will automatically download the latest ruleset configuration when the system starts on boot. However, a requirement of SecureNet Admin is the ability to 'Push' a new ruleset configuration to the client, thereby enforcing a newly defined configuration straight away. As a corollary, the design of SecureNet needed to ensure that when SecureNet Admin pushed a ruleset, there was a 'listener' to receive this update and update the local configuration file. This implies that the listener needs to be bound to a specific port on the

SecureNet computer, allowing the SecureNet Admin server to connect to this port and transfer the latest ruleset configuration.

5.2.4 Log Uploader

An additional subsidiary function of SecureNet is the ability to upload log data to the SecureNet Admin server. As web violation data will be different from chat violation data, the logs will be stored in separate files, and as such the frequency at which they are uploaded can differ.

By default, SecureNet will upload web logs every hour, whereas chat logs are uploaded immediately after the offence. Obviously, functionality to change this setting will be available in the ruleset configuration file. Consequently, SecureNet needs to know the IP address of the SecureNet Admin server, as well as the port to connect to in order to transfer the logs. Additionally, a mechanism is required to upload logs at a different frequency. The design for this mechanism is as follows:

- On start-up, start a threaded timer object.
- The timer should be able to produce events at different intervals, set the interval to one of the following (depending on the configuration file):
    - Minute ((almost) immediate log transfer)
    - Hour
    - Two hours
    - Every 24 hours (upload logs every day)
- When the timer event is fired, upload the specified log (if one is available) to the SecureNet Admin server.
- When the log has uploaded, clear the contents of the local log to prevent duplicate entries on the server log on the next upload.

5.3 SecureNet Admin Class Design

The second software component, SecureNet Admin, is responsible for displaying log data to an administrator, and also for defining rules in the configuration file. It is clear from the requirements that in order to satisfy the learnability and usability requirements, a graphical user interface (GUI) is essential in both instances to represent the relevant data.

The MVC paradigm is a way of breaking an application into three parts:

- Model – representing the nature and state of the visual objects
- View – the visible GUI responsible for presenting the data to the user
- Controller – the component allowing the model and view to communicate

We can base the design of SecureNet Admin using this paradigm, implementing the ruleset configuration file and log files as the model representing data elements, where as the GUI acts as the visible and controlling component, reacting to mouse and keyboard inputs from the user and mapping these commands to the model to effect the appropriate change.

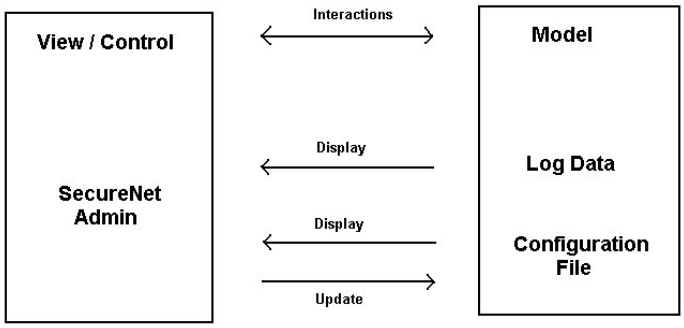

Figure 5.7 SecureNet Admin Design Structure

The design of the log and configuration files is trivial and discussion will be postponed until Section 6: Implementation.

5.3.1 SecureNet Admin GUI Design

The design of the SecureNet Admin application needed to ensure that the administrator could perform all administrative tasks including the browsing of both web access and chat log data, and the definition of configuration rules including the SecureNet system configuration, web filtering settings, chat monitoring settings, and user settings (encompassing both administrators and users).

In order to meet consistency and familiarity requirements, the main SecureNet Admin console screen will follow a template defining the basic appearance of the application:

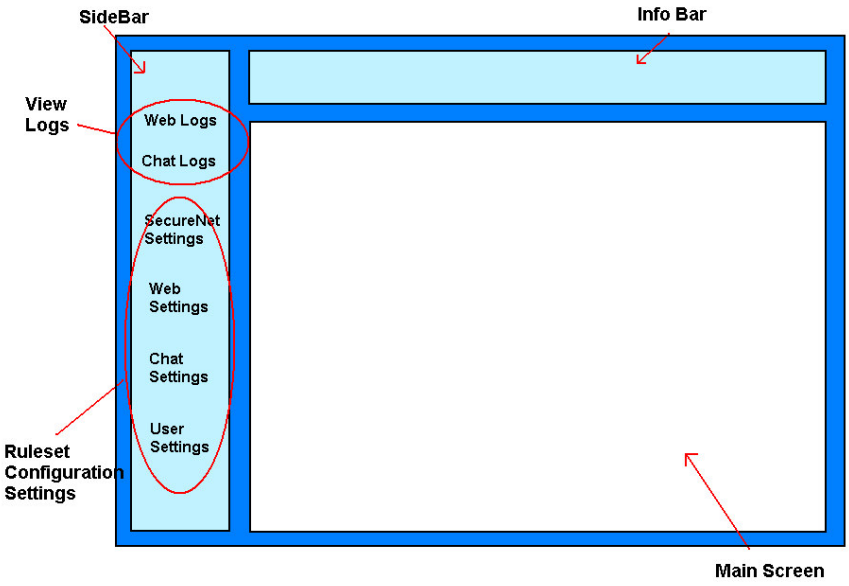

Figure 5.8 SecureNet Admin Console Screen Layout

The design of this template is closely based on the MSN 8 interface seen in Figure 3.1, and also on the Microsoft Outlook interface, imitating the use of the sidebar and information bar:

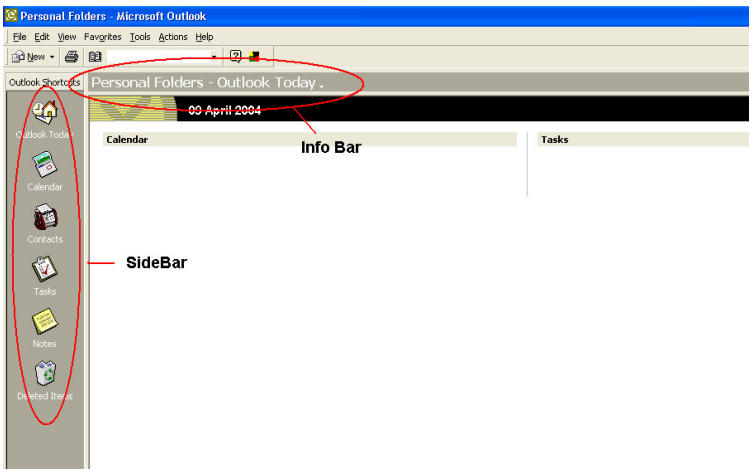

Figure 5.9 Microsoft Outlook Interface

The rationale behind this design is trivial:

- The SideBar and InfoBar remain constant, leaving only the Main Screen to change depending on the function chosen by the administrator.

- By default, the administrator will go to a 'Home' page, indicating the number of new logs that need viewing.

- The administrator can select to view logs or configure rule settings by clicking on the appropriate button on the SideBar. Since this is always present, the administrator can easily navigate between screens.

- The design of the SecureNet Admin console SideBar will incorporate icons in a similar fashion to Figure 5.9 to represent the functions provided, thereby imitating the look and feel of this screen to existing Windows software.

- The InfoBar is present to comply with the Synthesizability requirement, stating that the system must inform the user of the present system state. The InfoBar simply indicates which function the administrator is currently in, be it a view log function, or a configuration setting function.

- In order to comply with the familiarity requirement, the implemented colour scheme and icons will imitate Windows XP in style.

Further discussion of the icons that will be implemented is available in the 'SecureNet Logo and Icons' section.

5.3.1.1 Main Screen Design

The main screen can consist of one of any of seven screen-types depending on the functionality chosen by the administrator:

- Home screen – indicating the number of new violation logs

- Web logs screen – containing web access violation logs

- Chat logs screen – containing chat conversation violation logs

- SecureNet settings screen – containing configuration options for the SecureNet system

- Web settings screen – containing configuration rules defining allowed and blocked websites, and their respective logging options

- Chat settings screen – containing configuration rules for allowed/blocked chat applications, the definition of prohibited words and phrases, and their respective logging options

- User settings screen – allowing the definition of the security level that SecureNet is to operate at, the addition and editing of SecureNet Admin administrators, and also the addition and editing of SecureNet users, thereby allowing the definition of sensitive personal information into the system.

We will now discuss the design of the chat logs screen, chat settings screen, and user settings screen in further detail.

5.3.1.2 Chat Logs Screen

The chat logs screen is responsible for displaying chat conversation logs to the administrator. As defined in section 5.2.2, the screen will need to display the following information:

- User – the name of the user
- Contact – the email of the contact who the user is talking to
- Violation Type – indicates whether inappropriate language was used, or if sensitive information was released
- Keys Detected – the prohibited keys which were detected in the conversation
- Time – the time of offence
- Message – a transcript of the entire conversation

The design of this screen is closely based on that of Spector Pro (Figure 3.11), allowing the presentation of all of the log data simultaneously. Most importantly, the chat transcript can be seen in its own window, allowing easy viewing for the administrator. As specified, the screen contains the user, contact, violation type, keys detected and time in column format.

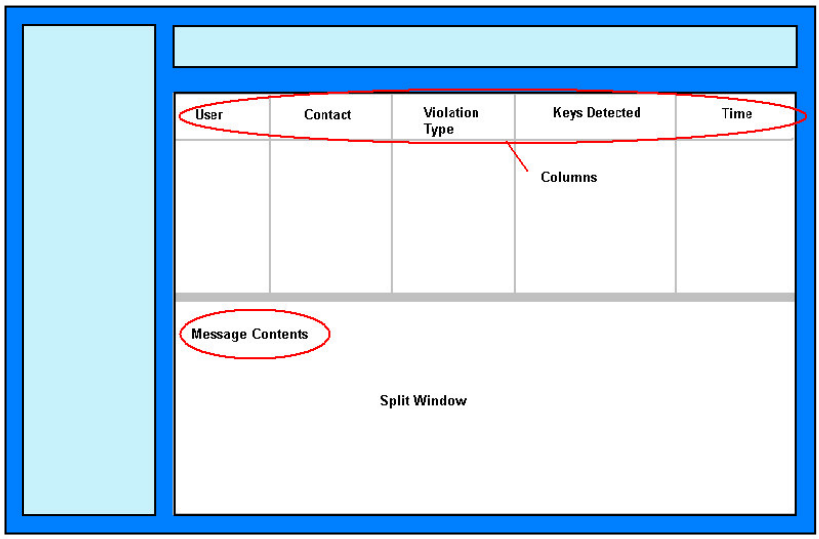

Figure 5.10 Chat Logs Screen

This format is repeated in the web logs screen with the exception of the split window displaying chat conversation transcripts.

5.3.1.3 Chat Settings Screen

The chat settings screen is responsible for allowing the configuration of permissible chat applications and the definition of prohibited words and phrases. In order to prevent functionality within a screen design from overcrowding, the SecureNet Admin screen design will make use of tabbed panes.

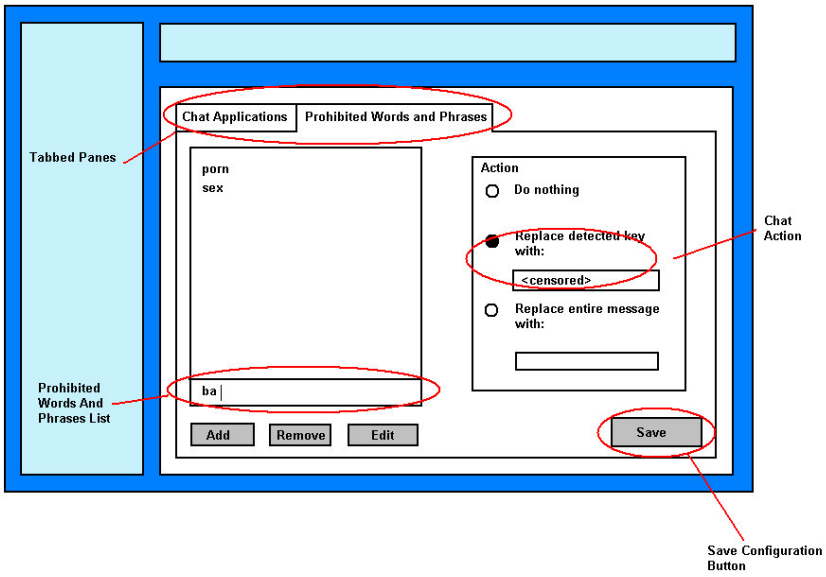

Figure 5.11 Chat Settings Screen

The tabbed-pane structure allows the administrator to flip through settings for the same function, but still have the ability to set a wide range of options without a single screen being over swamped with configuration options, making the screen difficult to comprehend and thereby limiting usability. Within the tabbed-pane screen, we have a prohibited words and phrases list, a structure repeated in the web settings screen allowing the definition of custom allowed and blocked websites. We can add to the list by typing in the word within the text box and clicking the 'Add' button. It is possible to edit or delete a word by selecting the word within the list and clicking on the respective button. To set the action to be taken on the detection of prohibited words, a group of radio buttons is present in the 'Action' panel. Replacement text can be defined within the given text boxes. Finally, the 'Save' button in the bottom right hand corner allows the updating and saving of the ruleset configuration file.

5.3.1.4 User Settings Screen

The final screen under discussion is the user settings screen, responsible for representing user data, including administrators within a tree structure. The user settings screen is responsible for holding data about system administrators, and entering sensitive personal data about system users. Although not implemented in the prototype, the tree structure can also be extended to allow the representation of groups, allowing different ruleset configurations to be applied. In the prototype system, this is restricted to defining the security level at which the web filter works at using a slide bar.

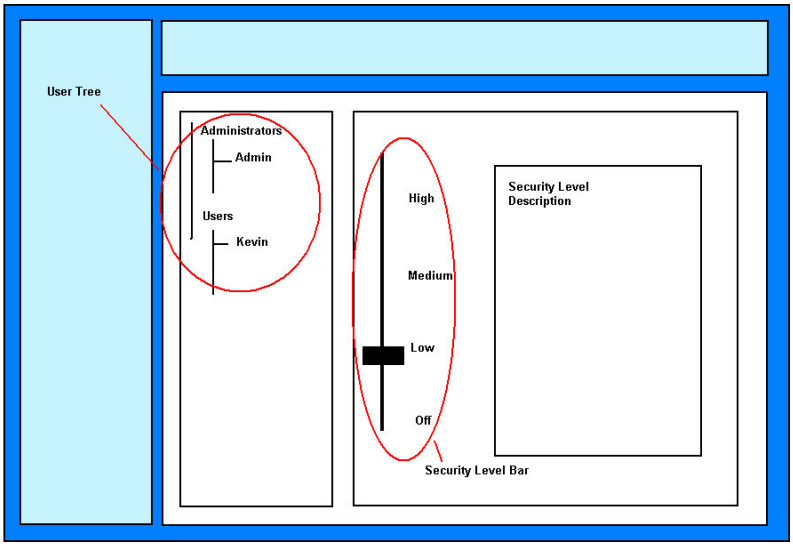

Figure 5.12 User Settings Screen

5.3.1.5 SecureNet Logo and Icons

As with any commercial application, the application logo allows users to identify the application they are using. The design of the SecureNet logo is trivial, and basically symbolises a key to 'lock up' the Internet from misuse. The 'Net' is in a slightly bigger font to represent the 'edges' of a key.

Figure 5.10 SecureNet Logo

Additionally, icons are used in the SideBar, enabling the symbolisation of the function. In order to comply with familiarity guidelines and give the application the Windows XP 'look and feel', icons were chosen from Sawyer's (2003) website Foood's Icons[16]. These icons were specifically chosen to draw on existing traditions and resemble the function for which they will symbolise:

Web log
Chat log
SecureNet

Web Settings
Chat Settings
User Settings
Configuration Settings

[16] Foood's Icons [online] http://www.foood.net/index.htm (Accessed 20/4/04)

6 Implementation

The purpose of the current chapter is to provide details of the implementation of key, non-trivial components of the system, highlighting some of the most important and innovative features of the implementation.

6.1 Storing Configuration File and Log Data

The implementation of the configuration file and storage of logs proved to be a surprisingly interesting aspect of system implementation. The trivial solution is to simply save both sets of data into a plain text file. However, this poses huge problems when we wish to edit the information or extract the data in a meaningful way. An option looked at in great detail was the use of XML, as C# language provides excellent facilities for its implementation. XML attributes could be used to store configuration settings in the following manner:

```
<ProxySettings>
        <Version value="1.0" />
        <Proxy set="false" ip="138.38.32.87" port="3128" />
        <BlockList>
                <Block IP="216.239.57.99" />
                <Block IP="129.35.76.220" />
                <Block IP="143.252.156.10" />
        </BlockList>
</ProxySettings>
```

Although this method was ideal for static data, that is the configuration of Boolean, int, or string values within an attribute, it was not as ideal for storing dynamic content such as the addition of extra websites to a custom blocked/allowed list, or the addition of log information to a file as XML does not provide a generic 'Append' method to add data directly to a node.

Instead we have to use a long-winded method that involved loading the XML document, searching through the document to find the desired node we want to append to, create a new element to hold attributes, and then append this child element to the node. In the case of log data where we have to save several attributes' worth of data, the consequent method to append data to a file quickly becomes very large. As a corollary, the size of the XML file that we are creating will also become very large, as it will hold all the additional attribute tags holding the information.

Another option is the implementation of 'serialisation', which enables the streaming of objects and its various member data to be written to a stream as a series of bytes. C# provides support for serialisation in the form of SoapFormatter and BinaryFormatter which allow the serialisation of objects into the SOAP protocol or into a binary stream respectively. Liberty (2003) defines SOAP as a simple, lightweight, XML-based protocol for exchanging data across the web. Although the use of SOAP would improve the portability of the code to different systems, it was decided that BinaryFormatting would be used on the rationale that

the use of SOAP would result in file sizes becoming too large, thereby causing both network congestion and the requirement of storing large logs on both the client and server machines.

In order to use serialisation, we must first create a library object of the serialised class (i.e. a template). This class is like any other class except we must add the [Serializable] attribute at the beginning of the source file to indicate that its data elements can be serialised. By creating a serialised library object, we can easily create configuration and log files holding both static and dynamic values through the use of readily available primitive types and data structures within the C# language. The library acts as a resource when we compile our main classes SecureNet and SecureNet Admin, allowing these classes to use the library as a template to save information by serialising data into a file, or extracting information by deserialising the data from the file. The code for creating serialised library objects for the ruleset configuration and log data storage can be found in Appendix C.

6.2 SecureNet

6.2.1 Implementing the Packet Analyser

As discussed in section 5.2.2, the core component required of SecureNet is a packet analyser, which would provide the filtering and monitoring engine. The prime implementational requirements were that the analyser had to capture packets transmitted between the web browser / chat-client, and the Internet, and be able to utilise configuration data to provide web filtering and monitor chat conversations for inappropriate content. Several options were considered during its implementation:

- Modification of existing packet analysers. Pacanal[17] is a C# packet analyser based on Ethereal. Like Ethereal, Pacanal utilises the freely available NDIS packet capture driver *'npf.sys'* developed by WinPCap[18] to capture packets sent between the local machine and the Internet. Unfortunately, investigation into the possibility of using this architecture resulted in disappointment after learning that as WinPCap was implemented as a protocol, it merely makes a copy of the packet, which can then be displayed within a GUI. As such, this method would be useless in creating a web filtering application as the architecture does not have the ability to drop packets or change their content.

- Implementation of a filter at NDIS level. NDIS stands for 'Network Driver Interface Specification', and a solution implemented at this level would provide filtering functionality at a low-level as NDIS works between the network device and the TCP/IP stack:

[17] Pacanal is available at http://www.codeproject.com/csharp/pacanal.asp (Accessed 1/5/04)
[18] WinPCap: Free Packet Capture is available at http://winpcap.polito.it/ (Accessed 1/5/04)

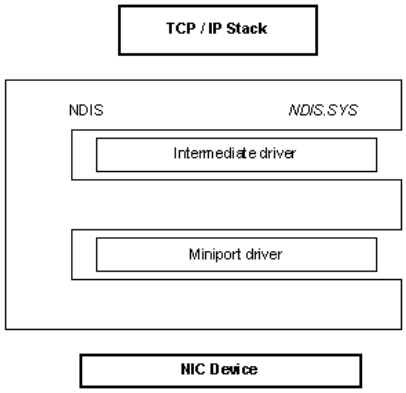

Figure 6.1 NDIS Architecture

After careful analysis of the feasibility of writing a NDIS driver[19] to carry out the desired functionality, it was decided that implementation of a solution at this level was infeasible given the time and resource constraints of the project.

- Implementation using a software-based 'proxy' architecture. This method relies on using a software-based proxy that acts as a 'hub' through which to direct Internet connection requests. As the proxy is located in the middle of the Internet connection, it will receive all incoming and outgoing packets, and can therefore be extended to analyse these packets to provide filtering and monitoring functionality.

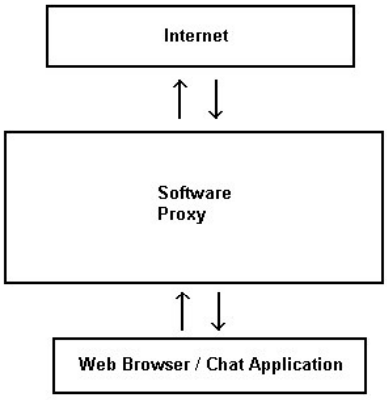

Figure 6.2 Proxy Architecture

Mentalis Proxy is a C# implementation of a software-based Http, Ftp, and SOCKS proxy server and provides an ideal platform on which to build our SecureNet packet analysing engine. A discussion of Mentalis Proxy is available in Appendix D.

[19] Writing an NDIS Driver:
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/network/hh/network/301int_78mf.asp
(Accessed 2/5/04)

By analysing the implementation of Mentalis Proxy, it was possible to rip out the core 'proxy' engine on which to build the SecureNet packet analyser:

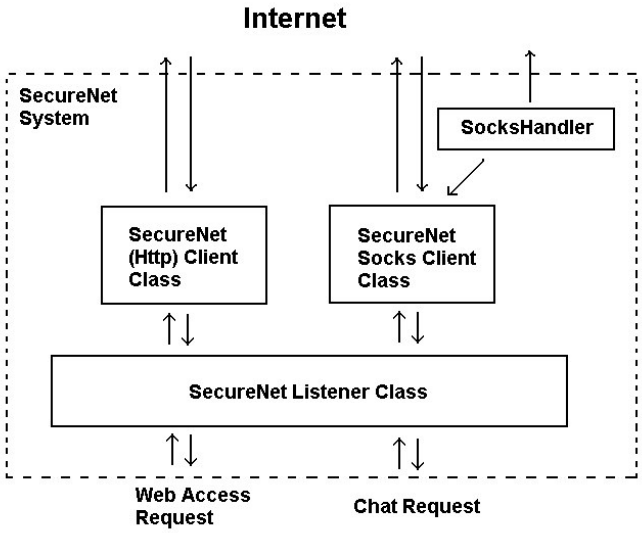

Figure 6.3 SecureNet Architecture

C# provides the use of 'Asynchronous Sockets' for network programming, a solution used extensively throughout the software system. The real significance of programming using asynchronous sockets is that network functions can be built to trigger events once they have completed, thereby passing program control to a delegate function to complete the task, and avoiding occurrences of application lock-up whilst waiting for the functions to complete. A full discussion on the intricate details of asynchronous sockets is beyond the scope of this document, although we will discuss in high-level detail, the way they work.

The Socket asynchronous methods split network functions into two parts:

- A Begin method that starts the network function and registers the AsyncCallback method.
- An End method that completes the function when the AsyncCallback method is called.

The methods we are most interested in are:

Asynchronous start method	**Description**	**Asynchronous end method**
BeginAccept()	Accepts an incoming connection	EndAccept()
BeginConnect()	Attempts connection to a remote host	EndConnect()
BeginReceive()	Retrieves data from a socket	EndReceive()
BeginSend()	Sends data to a remote socket	EndSend()

Depending on the type of application we are creating, we can either use BeginAccept() to listen for incoming connections to a pre-specified port, or BeginConnect() to connect to a pre-specified IP address and port. To quickly demonstrate their usage we will define an outline 'server' program which listens for connections, followed by an outline 'client' program which attempts connection to the server, and sends some information:

- Server

```
// define a new Socket object and bind it to port 9050
Socket server = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
                                                    ProtocolType.Tcp);
IPEndPoint iep = new IPEndPoint(IPAddress.Any, 9050);
server.Bind(iep);
server.Listen(5);

// start listening for a connection on this Socket
server.BeginAccept(new AsyncCallBack(OnAccepted()), server);

// Callback delegate for BeginAccept()
// When a connection request is received, we must explicitly End the
// BeginAccept() method, which in turn informs the client that their BeginConnect()
// method is successful.
// We then begin receiving data from the client.
void OnAccepted(IAsyncResult ar)
{
     //we know the details of the client Socket as we are passed the socket object
     //in the BeginConnect() request
     Socket client = server.EndAccept(ar);
     //start receiving data and stream data into byte[] Buffer
     server.BeginReceive(Buffer, 0, Buffer.Length, SocketFlags.None, new
                                  AsyncCallback(OnDataReceived()), server);
}

// Callback delegate for BeginReceive()
// After the data has been sent we must explicitly End theBeginReceive() method.
// This in turn informs the client that the send was successful.
// Finally we print the contents of the sent data to the console
void OnDataReceived(IAsyncResult ar)
{
     int Ret = server.EndReceive(ar);
     // if the size of the received data is 0, we assume that the client has
     // disconnected
     if(Ret != 0)
     {
               // print out the data received
               string message = Encoding.ASCII.GetString(Buffer, 0 , Ret);
               Console.WriteLine(message);
     }
}
```

- Client

```
// define a new Socket object
Socket client = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
                                                    ProtocolType.Tcp);
// define the address and port we want to connect to
```

```
IPEndPoint iep = new IPEndPoint(IPAddress.Parse("192.168.0.2"), 9050);
//attempt to connect to server at address 192.168.0.2, port 9050
client.BeginConnect(iep, new AsyncCallback(OnConnected()), client);

// Callback delegate for Connect
// when we have managed to connect to the server, the EndAccept() method
// triggers an event informing us that the connection is successful
// when we have connected, we send a message saying 'Hello'
void onConnected(IAsyncResult ar)
{
    client = (Socket)ar.AsyncState;
    try
    {
            client.EndConnect(ar);
            // send a message saying 'Hello'
            string  message = "Hello";
            byte[] msg = Encoding.ASCII.GetBytes(message);
            client.BeginSend(msg, 0, msg.Length, SocketFlags.None, new
                AsyncCallback(onSend()), client);
    }
    catch{} //we are unable to connect to the server
}

// Callback delegate for BeginSend()
// when the message is successfully sent, the server's EndReceive() method
// triggers an event informing us that we have sent the message successfully
void onSend(IAsyncResult ar)
{
    // end the send and close the socket
    int Ret = client..EndSend(ar);
    client.Close();
}
```

As previously mentioned, this network programming style is extensively used throughout the application and extracts can be seen in section 6.4. Alternatively, full listing is available in the supplied CD. For a more detailed discussion of asynchronous sockets please refer to C# Network Programming, Blum (2003).

6.2.1.1 SecureNet Listener Class

The listener class is responsible for listening for web access requests from the web browser and also for chat conversation connections. Additionally, it is also responsible for listening for ruleset configuration updates 'Pushed' by SecureNet Admin (to be discussed in Section 6.4). Given that the analyser will be working on the local machine, we set the proxy to listen on ports 80 and 1080 (Http and SOCKS respectively) on the loopback address '127.0.0.1'. We must then configure the web browser and chat client to use the proxy:

- Manually setting it:
    - Internet Explorer:
      Tools -> Internet Options; Connections Tab -> LAN Settings…;
      Select 'Use a proxy server'; Address: 127.0.0.1; Port: 80
    - MSN Messenger:
      Tools -> Options…; Connection Tab; Select 'I use a proxy server';
      Type: Socks 4 Server; Address: 127.0.0.1; Port: 1080

- Using the registry and changing the following keys:

    - Internet Explorer:
      HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings
      Set 'ProxyServer' to '127.0.0.1'; 'ProxyEnable' to '1'

    - MSN Messenger:
      HKCU\Software\Microsoft\MSNMessenger
      Set 'ProxyState' to 'ff ff ff ff'; Socks4Server to '127.0.0.1';
      Socks4Port to '38 04 00 00'

(In order to assist installation of SecureNet, the method for setting registry keys is included in the 'install.bat' file included with the application. The method can be viewed in the setreg.cs class in Appendix C).

When the listener class receives a request, it starts a new instance of either SecureNet Client or SecureNet SocksClient, which are responsible for dealing with Internet interaction of web access and chat conversations respectively.

6.2.2 SecureNetClient - Web Filtering

The SecureNetClient class is exclusively responsible for dealing with interactions between the web browser and the Internet. Given that the original Mentalis Proxy engine broke up the Http request into its constituent parts to check the URL address, implementing the URL filter algorithm defined in section 5.2.2 was reasonably trivial. What was more difficult was the creation, implementation, and application of the IP lists.

C# provides a class called ArrayList, allowing the creation of an array whose size is dynamically changed as required. This structure is ideal, as we want to hold a list of websites that will increase or decrease as the administrator adds/removes websites to/from the custom defined lists. Importantly, the ArrayList class includes the AddRange() method, allowing the addition of all the elements of another ArrayList, and also the Contains() method, allowing fast checking of whether or not a website is on the list.

As mentioned in Section 6.1, we will use a serialised class library template to define a structure to hold website lists. The SecureNet pre-defined list was created manually by creating an arraylist of websites appropriate for each category using the serialised library defined by SecureNetList (Appendix C). As this was a prototype, we limited the number of websites in each category to approximately 20 sites. For a full system, it is expected that the company would employ a team to create the lists professionally, checking them as needed and updating these lists for distribution to fee-paying customers. The following snippet of code demonstrates the implementation:

```
// define global ArrayLists
ArrayList AllowList = new ArrayList();
ArrayList BlockList = new ArrayList()

// load pre-defined list
string dir = Environment.CurrentDirectory;
dir += @"\SecureNet\SCN.lst";
```

```
FileStream strm = new FileStream(dir, FileMode.Open);

// deserialise the list
Iformatter formatter = new BinaryFormatter();
ArrayList list = (SecureNetList)formatter.Deserialise(strm);

// add websites to allow/block list depending on configuration setting
// 1=allow, 0=block, config =  configuration file

// foreach predefined category
if(config.SetAdultMatureList == 0)
        BlockList.AddRange(list.AdultMatureList);
if(config.SetAdultMatureList == 1)
        AllowList.AddRange(list.AdultMatureList);
```

Additionally, we had to combine this list with the administrator-defined custom list set in the configuration file. For ease of use, the administrator defines websites by URL name rather than by IP address, whereas our pre-defined lists are already in IP-address enabling fast comparison. As such, we need to define a method ResolveHosts() to perform a NSLookup of the defined URLs and retrieve all of their IP addresses:

```
// load custom-defined list from configuration file
string dir = Environment.CurrentDirectory;
dir += @"\SecureNet\scnconfig.cfg";
FileStream strm = new FileStream(dir, FileMode.Open);

// deserialise the list
Iformatter formatter = new BinaryFormatter();
Config config = (Config)formatter.Deserialise(strm);

// add allowed/blocked URLs to respective lists
AllowList.AddRange(ResolveHosts(config.CustomAllowedList));
BlockList.AddRange(ResolveHosts(config.CustomBlockedList));

// perform nslookup on URLs
private ArrayList ResolveHosts(ArrayList list)
{
        ArrayList hosts = new ArrayList();
        // foreach custom defined URL
        foreach(string host in list)
        {
                // perform NSLookup
                IPHostEntry results = DNS.GetHostByName(host);

                // add each ip address registered to the domain name to the list
                foreach(IPAddress address in results.AddressList)
                {
                        hosts.Add(address);
                }
        }
        // return list of ip addresses
        return hosts;
}
```

We now have a fully populated AllowList and BlockList with which we can do our filter checks, as defined by our filtering algorithm in section 5.2.2.

6.2.3 SecureNetSocksClient - Chat Monitoring

The SecureNetSocksClient class is principally responsible for dealing with interactions between the MSN Messenger chat client and the Internet, delegating the SOCKS protocol interaction to the SecureNetSocksHandler class. The original Mentalis Proxy engine provided functionality to establish a SOCKS connection with an application server (see Appendix D) after which transmissions were relayed between the remote server and local client using the StartRelay() method. We can implement a packet analyser at this point to inspect the contents of relayed packets for chat conversation based on the MSN Messenger protocol.

As discussed in section 5.2.2, before a chat conversation can commence, a packet with the 'JOI' or 'IRO' command must first be sent in order to establish a chat conversation. As such, we can design a function IsMSNMessage() to specifically look out for these commands and inform us whether the contents of the packet should be logged. Because C# networking involves the transfer of data using byte arrays, it is possible to extract the contents of this buffer into a string by using the following code:

```
String message = Encoding.ASCII.GetString(buffer, 0, ret);
```

where message is the extracted data using ASCII encoding, buffer is the byte array, 0 is the starting position of the array to read from, and ret is the number of bytes of data we have to read – as informed by the AsyncCallback method. Now we have a string representation of the packet, we can use the StartWith() method which allows us to inspect whether a specified string starts with the specified characters:

```
if(message.StartsWith("JOI") || message.StartsWith("IRO"))
{
        // we have identified the start of a conversation
        // parse the packet header to extract information about the contact's email
        // address and alias for our log file
}
```

Once we have detected the start of a conversation, the next step is to look for the double newline sequence '0d 0a 0d 0a'. We can search for this sequence using the IndexOf() method, which will search for a specified string, and return the index of the string if it is found. If the sequence cannot be found, we can conclude that the given packet is not a conversation packet, otherwise we test the length of the packet against the index at which the sequence is found. Remember that a 'typing user' packet terminates with this sequence whereas a packet containing a chat message will have characters after this sequence:

```
// find double newline sequence and return its index
int index = message.IndexOf("\r\n\r\n");
int length = message.Length;

// sequence not detected
if(length == -1)
{
        return false;       // packet does not contain conversational data
}
```

```
// sequence detected but there is no content after it
if(index == length)
{
        return false;       // packet does not contain conversational data
}

// else sequence detected – parse the packet into its constituent parts using the
// ParseMSN() method
StringDictionary MSNService = ParseMSNPacket(message);
```

Another feature provided by C# is the StringDictionary class which provides a hashtable data structure, with the key strongly typed to be a string rather than an object. This structure proved to be very valuable when parsing an MSN packet because it allowed the packet field 'headers' i.e. MIME-Version, Content-Type, TypingUser/X-MMS-IM to be used as keys, whilst their content is stored as the value. As such we can create a very fast and efficient method ParseMSNPacket() to parse out the content of the packet and retrieve it where necessary:

```
private StringDictionary ParseMSNPacket(string query)
{
        // define a new StringDictionary object
        StringDictionary retdict = new StringDictionary();

        // first we extract each line separated by a newline '\r\n'
        // sequence into an array
        string[] Lines = query.Replace("\r\n", "\n").Split('\n');

        // then we parse the packet header. As each element is
        // separated by a white space we simply find the index
        // of this space and then use the Substring method()
        int ret = Lines[0].IndexOf(' ')
        MSNCommand = Lines[0].Substring(0, ret);

        // trim whitespace before and after remaining string
        Lines[0] = Lines[0].Substring(ret).Trim();

                //continue parsing header until we have parsed each element

        // next we extract the packet fields into the string dictionary
        for(int cnt=1; cnt <4; cnt++)
        {
                // look for ':'
                ret = Lines[cnt].IndexOf(':');

                // substring before ':' is the key, everything after its value
                retdict.Add(Lines[cnt].Substring(0, ret), Lines[cnt].Substring(ret+1));

        }

        // if we have conversational content
        if (Lines.Length > 4)
        {
                MSNMessage = Lines[5];
        }
        return retdict;
}
```

Now that we have parsed the packet, we can do an easy check to see whether the present packet has conversational content:

```
// We already know that a sent message has content type: text/plain; charset UTF-8
if(!MSNService[“Content-Type].Equals(“text/plain; charset = UTF-8)
{
        return false;
}
else
{
        return true;        //conversation found
}
```

Incidentally, the IsMSNMessage() method also looks out for the ‘OUT’ command, signaling the end of a conversation.. When this is seen, the method will save the logged data to file depending on the configuration setting.

The next requirement we need to implement is the ability to scan conversations for violations such as language of a sexual or abusive nature, or the release of sensitive details. As we have parsed the conversation into a readily accessible string, we can use another C# class Regex to match for violation keys within our conversation string. Regex is a specialised regular expression class provided by .Net as an object-oriented approach to regular expression matching and replacement. As such, we can also apply the use of this class to our requirement of censoring any violation keys found:

```
bool ViolationDetector(string message)
{
        bool found = false;

        // prohibitlist is our list of words and phrases deemed inappropriate from the
        // configuration file
        foreach(string key in prohibitlist)
        {
                // match() method attempts to match any occurrence of key found in
                the message
                Match violation = Regex.Match(message, key);

                // if a match is found, set found to true
                if(violation.Success == true)
                {
                        found = true;
                }
        }

}
```

Note that the method for replacing a found key with a <censor> message is very similar to that outlined above, with the exception that the Replace() method is used in place of Match(), shown as follows:

```
String newMessage = Regex.Replace(message, key, censor_string);
```

The last thing we need to do now is rebuild the packet with the censored message. As we have already parsed the original message into its constituent pieces, 'gluing' it back together is a very trivial matter as all we have to do is create a new empty string and add the original pieces back to, with a 'few' changes where we have censored content:

```
string RebuildMessage()
{
        string newMsg = "";
        newMsg += "MIME-Version:"  +(string)MSNService["Mime-Version"] + "\r\n";
        newMsg += "Content-Type:"  +(string)MSNService["Content-Type"] + "\r\n";
        newMsg += "X-MMS-IM-Format:"  +(string)MSNService["Mime-Version"]
                                                        + "\r\n";
        newMsg += "\r\n";

        string censoredMsg = "";
        // censoredMsg = censored message

        newMsg += censoredMsg;
}
```

Additionally, we must calculate and add the new length of our newly created packet, otherwise MSN Messenger will over- or under-display text depending on whether the new length is longer or shorter than that of the original packet:

```
int msgLength = newMsg.Length;
MSNMsgLength = msgLength;

// create the header
String header = MSNCommand + MSNEmail + MSNAlias + MSNMsgLength;

// create the new packet
header += newMsg;
```

Further details of the SecureNet class can be found in the SecureNetClient folder of the submitted CD.

6.2.4 Implementing Transparency

In order to comply with the requirement of SecureNet operating transparently on the client computer, we needed to ensure that SecureNet could run in the background without any visible interface or screens on the desktop. Windows XP 'Services' allow programs to run as processes in the background, and conveniently, Visual Studio .Net provides tools which we can trivially utilise to implement SecureNet as a 'Service'.

Using Visual Studio .NET, we can use a template called 'Windows Service' that automatically creates a new Windows Service class which can then be edited to provide the functionality required. The steps taken to customise this for SecureNet are as follows:

- Within the InitialiseComponent() method, we changed the ServiceName property to 'SecureNet'. This allows the definition of the service name shown in the Windows Services list.
- Within the OnStart() method, we started the listener objects for Http, Socks and SecureNet Admin connections.

- Within the OnStop() method, we disposed of the listener objects for Http, Socks and SecureNet Admin connections.

- Additionally, we added a threaded timer object which allows the periodic update of logs to the SecureNet Admin server (see section 6.4 for more details).

To install the service, the installutil service installer must be executed from the command line, with the output of the service class (SecureNet.exe) as an argument. The service must then be started using the net start command. This has been implemented in the install.bat file which aids SecureNet installation.

Conversely to the idea of operating transparently, the requirements do stipulate that users should be informed that they are being monitored on the rationale that it acts as a deterrent and also informs users of the guidelines of acceptable computer use. As such we need to provide some warning message to the users. We can do this by displaying a message before logon, stating the SMART rules as defined by the Internet Watch Foundation (2003), by setting the following registry key:

HKLM\Software\Microsoft\Windows NT\CurrentVersion\WinLogon

And defining the following keys:

- Forceunlocklogon – forcing the user to read the message before logon
- LegalNoticeCaption – the title for the warning dialog box
- LegalNoticeText – the warning text

Again, this process is carried out as part of the install.bat program. The code required can be found in setreg.cs in Appendix C.

6.3 SecureNet Admin

As discussed in section 5.3, the second software component, SecureNet Admin, is responsible for displaying log data to an adminstrator, and also for defining rules in the configuration file. A graphical user interface (GUI) was an essential requirement for representing the relevant data. One of the primary factors for choosing C# and the .Net Framework as the development platform for this project was the provision of Windows Forms, allowing the creation of feature-rich Windows applications. Combined with an Integrated Development Environment (IDE) such as Visual Studio .Net (VS.Net), it is possible to rapidly create the overlying GUI using the front-end development tools.

VS.Net aids the development process by allowing the visual development of Windows Forms applications, allowing drag-and-drop functionality to place controls (visible component) directly onto the form from the toolbar window and manipulating its properties within the properties window. As such, this process cuts down on a huge amount of development time as the code required to develop the GUI is automatically generated, leaving the programmer to deal with the corresponding events which occur when a control such as a button is clicked, and with the loading and saving of data.

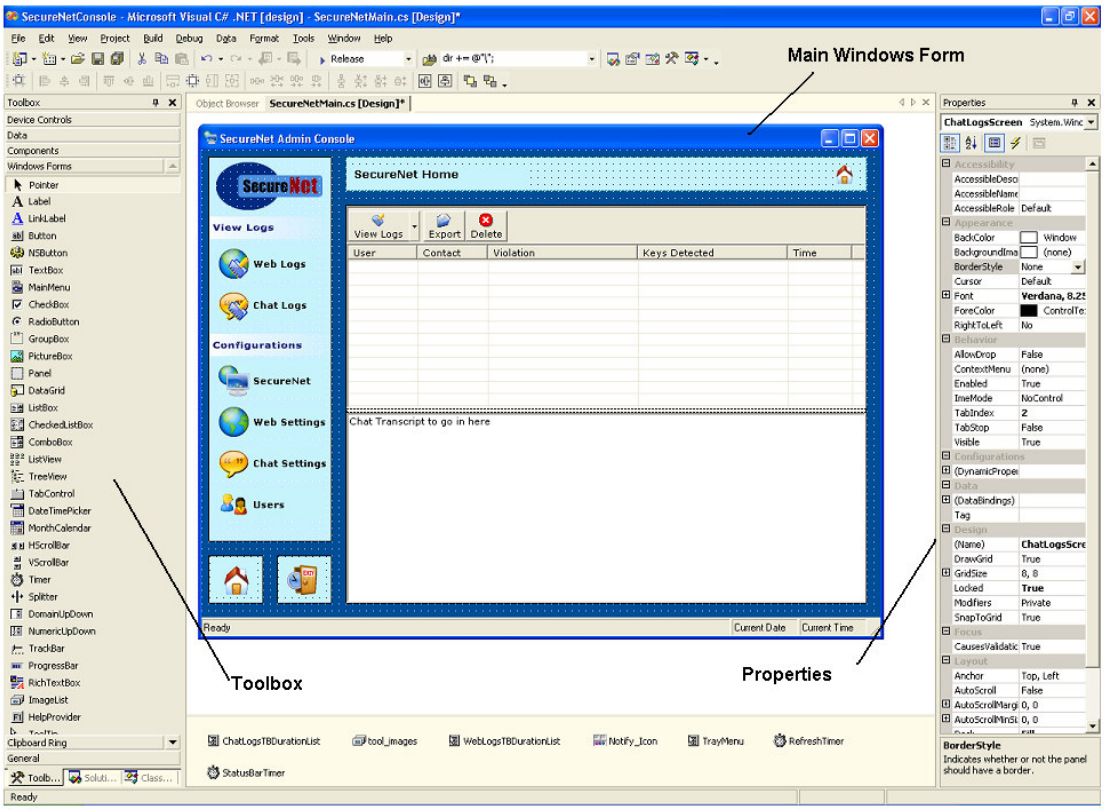

Figure 6.4 The Visual Studio .Net Development Environment

6.3.1 System Logon

As defined in the requirements section of this document, the SecureNet Admin console should have access controlled through the use of a login dialog, requiring a correct username and password combination. Surprisingly, although login forms are included as part of .Net's Web Forms package, there isn't a login template for normal C# applications. As such, we were required to program our own, which proved to be an interesting topic for discussion.

Initially, the login form had to be designed through the main VS.Net window, basing the design on existing Windows logon dialog boxes. Fortunately, the textbox control does provide properties to mask the entered password, available through the properties toolbox. As we weren't using a backend database for the application, we had to create our own administrator files holding username and password details. Again, we used a serialised library template to create these files (see Credentials.cs in Appendix C). To provide an extra layer of security, we

could utilise the Cryptography library in .Net to provide encryption using the triple DES encryption algorithm as follows:

```
// assume we have already filled in admin details into a serialised
// credentials object 'cred'

//create encryption keys (these are basic for proof-of-concept)
byte[] Key = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x10, 0x11,
0x12, 0x13, 0x14, 0x15, 0x16};

byte[] IV = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x10, 0x11, 0x12,
0x13, 0x14, 0x15, 0x16};

//serialise data into memory stream
MemoryStream memstrm = new MemoryStream();
IFormatter formatter = new BinaryFormatter();
formatter.Serialize(memstrm, cred);

//initialise triple DES encryption service
TripleDESCryptoServiceProvider tdes =
                    new TripleDESCryptoServiceProvider();

//create a filestream to save data
string dir = Environment.CurrentDirectory;
if (!dir.Substring(dir.Length - 1, 1).Equals(@"\"))
        dir += @"\AppData\";

//add username to string
dir += UserSettings_AddAdminUsernameTB.Text + ".scn";

Stream str = new FileStream(dir, FileMode.Create, FileAccess.ReadWrite);
//create encryption stream
CryptoStream csw = new
        CryptoStream(str,tdes.CreateEncryptor(Key, IV), CryptoStreamMode.Write);

//get bytes from memory buffer and encrypt its contents to file
byte[] cryptdata = memstrm.GetBuffer();
csw.Write(cryptdata, 0, cryptdata.Length);
```

To check whether a given username and password are valid, we have to check each file by iterating through all the files within a directory, deserialising and decrypting their contents until we found a match:

```
// populate an array with all admin files
String[] users = Directory.GetFiles(dir, "*.scn");
bool found = false;

// foreach file, deserialise and decrypt and check the contents against the
// given username and password
foreach (string file in users)
{
        Stream strm = new FileStream(file , FileMode.Open);

        //create encryption keys
        byte[] Key = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x10,
        0x11, 0x12, 0x13, 0x14, 0x15, 0x16};
```

```
        byte[] IV = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09,0x10,
        0x11, 0x12, 0x13, 0x14, 0x15, 0x16};

        //initialise encryption service
        TripleDESCryptoServiceProvider tdes = new
                                                TripleDESCryptoServiceProvider();

        //initialise decryption stream and memory stream
        CryptoStream csr = new CryptoStream(strm, tdes.CreateDecryptor(Key, IV),
                                                        CryptoStreamMode.Read);
        MemoryStream memstrm = new MemoryStream();

        //decrypt file to memory stream
        strm.Position = 0;
        byte[] data = new byte[1024];
        int recv = csr.Read(data, 0, data.Length);
        memstrm.Write(data, 0 , recv);
        memstrm.Position = 0;

        //deserialise decrypted stream
        IFormatter formatter = new BinaryFormatter();
        Credentials cred = (Credentials)formatter.Deserialize(memstrm);

        //check user credentials against the given username and password
        if((cred.UserName.Equals(username)) &&
                                (cred.Password.Equals(password)))
        {
                found = true;
        }
}
```

If a match is found, we can then grant access into the main administration console. Further details of this process can be found in the SecureNetConsole\Login.cs file in the CD.

6.3.2 The SideBar

As mentioned in section 5.3, the application uses a template of the sidebar and infobar, keeping screen navigation consistent between screens. The sidebar is a key component responsible for allowing the user to select different screens, whereas the infobar is responsible for informing the user of the current screen.

In order to make the buttons ‘work’, we have to implement them to respond to events such as a user clicking on a button. This is relatively straight-forward to do using VS.Net, as double-clicking on the control in the main window will automatically register the event-handlers for the control component leaving the developer to implement the actions to take upon the event within the event handler:

```
// Event handler for WebLogs button on the side bar
private void WebLogs_Click(object sender, System.EventArgs e)
{
        // we enter in this code to make the button ‘do something’
        this.HeaderLbl.Text = "Web Logs";
        ShowHeaderPic(this.WebLogPic);
```

```
        ShowPanel(this.WebLogsScreen);
}
```

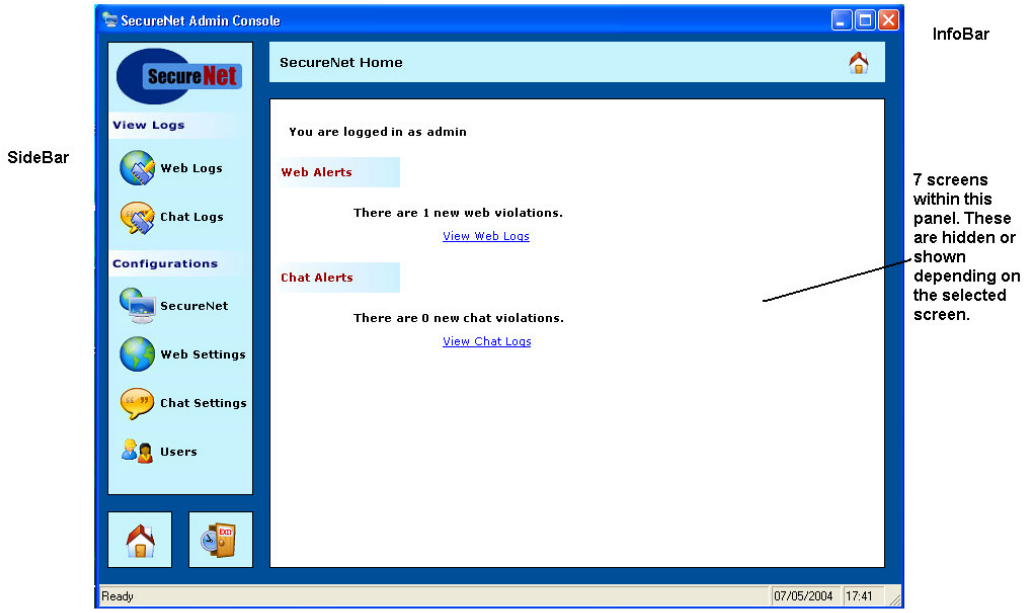

Figure 6.5 SecureNet Admin Layout

The main screen consists of a 'panel' on which other panels are 'held'. In total we have 7 further panels held within the 'main' panel, one for each screen. In order to display the appropriate screen, we implemented a simple function ShowPanel () which was responsible for brining the 'in focus' screen to the front and hiding all the other screens:

```
// Method to show active main panel
private void ShowPanel(Panel paneltoshow)
{
        foreach(Panel panel in panels)
        {
                panel.Hide();
        }
        paneltoshow.Show();
}
```

6.3.3 Displaying Log Data

The first of two primary functions is to display log data of violations uploaded from the SecureNet system. In order to display these logs, we use the ListView control which allows us to display data inside a table, commonly seen in utilities such as Windows Explorer. We won't discuss loading log data, as the code required was substantial given the amount of things we are logging. Information regarding log loading can be found in the SecureNetConsole\SecureNetMain.cs file in the submitted CD.

The following extract of code illustrates how a weblist is populated using the data from our logs:

```
//create an array of logs
Object[] webactivity = WebList.List.ToArray();

// foreach log in our array
foreach(URLCapture capture in webactivity)
{
        //parse fields
        string user = capture.User;
        string url = capture.URL;
        string ip = capture.IP;
        string action = capture.Action;
        string time = capture.Time;

        // enter each field into the list view by creating a new
        // listviewitem and passing it a string array of our parsed fields
        ListViewItem item = WebLogsView.Items.Add (
                new ListViewItem (
                        new string[] {user,url, ip, action, time } ));
}
```

The chat logs screen adds to the web logs ListView functionality by adding a split pane to allow the display the chat transcript in another window. Its implementation is trivial, as within the screen we simply dock the original listview control we had in the weblogs screen (with the obvious changes) and then dock it to the top of the chat logs screen. We then add a splitter control which allows resizing of the split pane. Finally we add a text box in the lower half of the chat logs screen and dock that to the bottom of the screen. We now have a split pane:

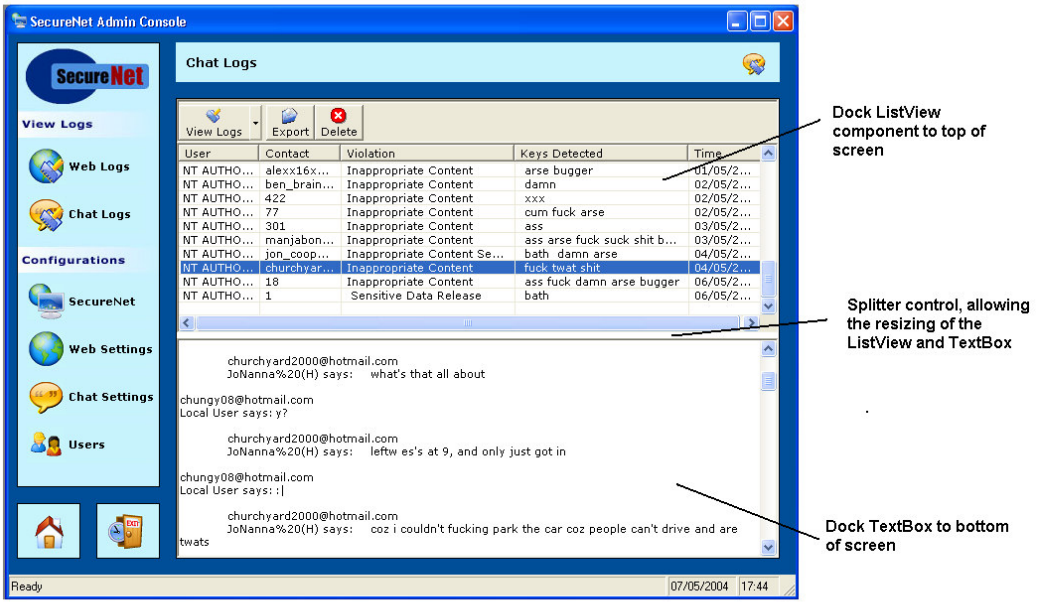

Figure 6.6 SecureNet Admin, Chat Logs Screen

A toolbar is included with both log views to provide functionality to filter logs depending on their age, together with a log deletion button and an export logs button to allow an admin to delete or export a file respectively. The export file functionality allows extraction of the log data to a text file format which can then be printed or emailed if required.

One thing not implemented was the ability to sort each row by clicking on the column. Although this was a requirement, time pressures resulted in its implementation being infeasible, as we would have had to implement a sorting function for each of the columns thereby creating a sizable task.

6.3.4 Ruleset Configuration

The second of the primary functions is to allow the configuration of SecureNet remotely. It was decided to split configuration options into 4 sections: SecureNet system, Web Settings, Chat Settings and User Settings to avoid screen clutter. The standard layout is to use a series of tabbed-panes that can be toggled between by the administrator. Within the tabbed panes, we then create a series of controls such as combo boxes, check boxes, list boxes with which the administrator can define ruleset configuration.

An interesting aspect of the implementation was the use of the treeview control to display system users. Although the requirements stipulated that the implementation of user groups was unnecessary due to constraints, the system still required an area to define sensitive personal data. It was felt that the best way to input such data would be to create ‘users’ which would hold this information.

To add a node to the treeview, we first had to decipher which node we wanted to add to. In the instance of adding a user to the ‘Users’ root node, we must first ensure that the user has selected the correct node before we can add a user:

```
// checking the parent node
if(this.UserSettings_UserTree.SelectedNode.Parent.Text.Equals("Users"))
        // display the add user settings screen

// wait for the user to type in user details

// enter this data into a serialised ‘users’ file then add the user to the tree
data.FirstName = this.UserSettings_AddUserFirstnameTB.Text;
data.Surname = this.UserSettings_AddUserSurnameTB.Text;
data.Address = this.UserSettings_AddUserAddressTB.Text;
data.PhoneNum = this.UserSettings_AddUserPhoneTB.Text;
data.MobileNum = this.UserSettings_AddUserMobileTB.Text;
data.SensitiveInfo =
 this.UserSettings_AddUserSensitiveTB.Text;

//add node to tree
UserSettings_UserTree.Nodes[1].Nodes.Add(user);
```

}

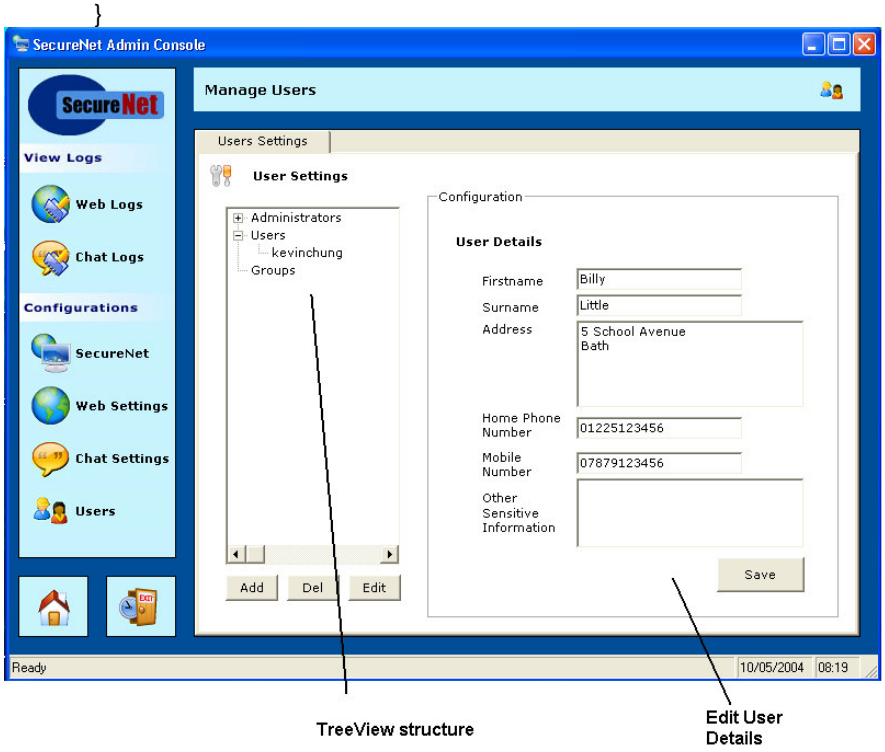

Figure 6.7 SecureNet Admin, User Settings

With hindsight into future work, this implementation can easily be expanded to include user groups, and as such it would be possible to move users into a 'group' using the treeview structure.

6.3.5 System tray application

There was a requirement for the application to be unobtrusive to normal computer activities, as such, the application was made into a system tray application, freeing up the desktop and taskbar, and only alerting the administrator by flashing the icon rather than by pop-up windows on the rationale that pop-up windows are irritating.

To make the application appear in the system tray is very trivial: simply drag a NotifyIcon from the forms designer in VS.Net into the application and set its properties to give it the desired icon and text. In order to make the application window disappear from the task bar, we must override the application's onClosing windows form event and cancel it:

```
// override form_Closing event
private void SecureNetAdmin_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
{
        // Hide the application
        this.Hide();

        // Cancel the close
        e.Cancel = true;
}
```

The application by default checks for new logs every 45 seconds so that the administrator can be alerted as soon as new logs arrive. If new logs are detected, it will alert the administrator by flashing the system tray icon. To animate the system tray icon, we use a timer object to produce events at 0.4 second intervals, and at each event, we show the icon, or hide the icon depending on its previous state, thereby giving a flashing appearance:

```
// create a timer object
flashtimer = new Timer();
flashtimer.Interval = 400;
flashtimer.Enabled = true;
flashtimer.Tick += new System.EventHandler (OnTimerEvent);

// timer event handler
void OnTimerEvent(object source, EventArgs e)
{

        if(m_IconVisible == true)
        {//hide icon if viewable
                Notify_Icon.Icon = m_App_Off;
                m_IconVisible = false;
        }
        else
        {//show icon
                Notify_Icon.Icon = m_App;
                m_IconVisible = true;
        }
}
```

When we log back in, we have to restore the application, and also have to reenter login credentials. If the credentials are valid we have to return the application back to the normal state and use the Show() method to make the application appear on screen:

```
// show the main screen
this.WindowState = FormWindowState.Normal;
this.Show();
```

Due to the size of the code of SecureNet Admin (10000+ lines), the full program listing can be found at SecureNetConsole\SecureNetMain.cs on the attached CD. A full set of screen captures of SecureNet Admin can be found in Appendix E.

6.4 Client-Server File Transfer

We discussed in high-level in section 6.2.1, that the procedure used to program network applications in C# differs depending on whether the application performs the role of a server or a client. In our case, both applications had to perform both roles to some extent. In the first case, the client had to continuously listen for configuration updates, and as such, it plays the

role of the server, whilst when it needs to upload a file, it is quite obviously the client. The same is true for the SecureNet Admin application, as when it is in receive mode to receive logs, it is the 'server', yet when it pushes out configuration files to SecureNet, it is obviously the client. As such there is much overlap between the two, allowing us to discuss in further detail the overview of the client-server networking process, and detail how file transfer was implemented in the solution.

We will discuss the process of configuration file transfer from SecureNet Admin to SecureNet:

By default, SecureNet is configured to perpetually receive configuration updates on port 9779. As such, it implements a listener method, using the BeginAccept() asynchronous socket method to await a connection and acts as the 'server' in our overview. When an administrator wants to send an update, SecureNet Admin attempts to establish a connection using the BeginConnect() method. Once established, the first step we take is to calculate the size of the file we are about to send, rather than send the file first. The rationale behind this method is that it prevents a partial file from being sent and then deserialised by the receiving party, thus causing an error. We implement this as follows:

```
// end the connect request when our connection request has been accepted
RemoteSocket.EndConnect();

//open log file
FileStream strm = new FileStream(dir, FileMode.Open);
Long size = strm.Length;

// create a new byte array
byte[] datasize = new byte[4];

// assign the value of this array to be the size of the file we wish to send
datasize = BitConverter.GetBytes(size);

//send data on log size
RemoteSocket.BeginSend(datasize, 0, 4, SocketFlags.None,
                        new AsyncCallback(SendFile), RemoteSocket);
}
```

In the SecureNet program, once we accept a connection, we start the BeginReceive() method in anticipation of receiving some data. As we designed the procedure, we know that the first piece of data will be a byte array declaring the size of the data file we are about to receive. As such, we can create a new byte array, and set its size to that required for the incoming file:

```
int Ret; = RemoteSocket.EndReceive(ar);

//set size of filebuffer for incoming file
int size = BitConverter.ToInt32(Buffer,0);
FileBuffer = new byte[size];

//start receiving configuration file
RemoteSocket.BeginReceive(FileBuffer, 0, FileBuffer.Length, SocketFlags.None,
                    new AsyncCallback(this.OnDownloadComplete), RemoteSocket);
```

In the SecureNet Admin program, we know that when the EndReceive() method is called, they have received the initial message declaring the size of the file. We can now send the configuration file:

```
int sent = RemoteSocket.EndSend(ar);

//open config file
FileStream strm = new FileStream(dir, FileMode.Open);
Long size = strm.Length;
Byte[] data = new byte[size];
Int recv = strm.Read(data, 0, data.Length);

//send data
RemoteSocket.BeginSend(data, 0, recv, SocketFlags.None,
                        new AsyncCallback(FileSent), RemoteSocket);
```

Finally, SecureNet has to receive this file and save it as the new configuration file:

```
//save config file to current directory
int Ret = RemoteSocket.EndReceive(ar);

//get user data directory
string dir = Environment.CurrentDirectory;
if (!dir.Substring(dir.Length - 1, 1).Equals(@"\"))
        dir += @"\SecureNet\";
dir += @"scnconfig.cfg";

FileStream writer = new FileStream(dir, FileMode.Create, FileAccess.ReadWrite);
writer.Write(FileBuffer, 0, Ret);
writer.Close();

//shut down remote socket
RemoteSocket.Close();
```

This client-server architecture is repeated several times (in slightly different ways) in the process of uploading log files and requesting configuration files within the system. A full listing of the described process may be found in the SecureNetConsole\SCNUpdateHandler.cs and SecureNetClient\SecureNetListener.cs files in the attached CD.

7 Testing

The purpose of this chapter is to discuss the tests performed on the resulting software of this dissertation. Each component was put through black box testing to ensure that the system conformed to the system requirements defined in chapter 3. The testing process involved the definition of test cases based on system requirements, and the expected output. The results can be seen in Table 7.1 and 7.2 for SecureNet and SecureNet Admin respectively.

Requirement	Action / Sub-Requirement	Expected Output	Actual Output	Pass/Fail
Filter URL Access, Security Level = High	Access a custom-defined 'Allowed' URL	Allow Access	Access Permitted	Pass
	Access any other URL	Block Access – Display Blocked URL Page	Displayed Blocked URL Page	Pass
Filter URL Access, Security Level = Medium	Access a URL in predefined category set to 'Allowed'	Allow Access	Access Permitted	Pass
	Access a custom-defined 'Allowed' URL	Allow Access	Access Permitted	Pass
	Access any other URL	Block Access – Display Blocked URL Page	Displayed Blocked URL Page	Pass
Filter URL Access, Security Level = Low	Access a URL in predefined category set to 'Blocked'	Block Access – Display Blocked URL Page	Displayed Blocked URL Page	Pass
	Access a custom-defined 'Blocked' URL	Block Access – Display Blocked URL Page	Displayed Blocked URL Page	Pass
	Access any other URL	Allow Access	Access Permitted	Pass
Filter URL Access, Security Level = Off	Access any URL	Allow Access	Access Permitted	Pass
Monitor Chat Conversations	Records an entire chat conversation if an inappropriate key is detected.	Produces a log with transcript of entire chat conversation	Log of chat conversation is made	Pass
	Censor key when detected	Censors keys within a message	Violating keywords are censored	Pass
	Censor entire message when key is detected	Censors entire message	Entire message censored	Pass

Requirement	Action / Sub-Requirement	Expected Output	Actual Output	Pass/Fail
Logging	Log everything	Logs all chat and all web violations	Logs all web access. Logs all chat conversations	Pass
	Log violations only	Logs web or chat where violations are detected	Logs only instances where violations are detected	Pass
	Log nothing	No logging is performed	No logging	Pass
	Upload logs to SecureNet Admin server	Successful upload of logs to server	Uploads logs to server	Pass (**)
Transparency	Application should be unnoticeable to the user	Little or no delay in web access times or chat conversation.	No noticeable lag in browsing websites or chatting	Pass
		No evidence of application on desktop or taskbar	Application completely hidden as it runs as a service	Pass
Information	Inform user of acceptable use guidelines	Acceptable use dialog at logon	Warning message shown prior to login	Pass

Table 7.1 Test Results: SecureNet

Requirement	Action / Sub-Requirement	Expected Output	Actual Output	Pass/Fail
Password Protection	Grant access to console on entering correct username and password	Access to SecureNet Admin console	Access permitted on giving correct username and password otherwise denied	Pass
	Create new administrators from within console	Additional administrators allowed access to SecureNet Admin	New administrator username and password allowed access	Pass
Logging	Allow continuous retrieval of logs	New logs which are uploaded immediately should be retrieved within 2 minutes	Allows continuous retrieval and update of logs.	Pass
	Application should be unnoticeable to the user	Application appears in the system tray	Application appears in system tray, hiding the application from the taskbar. Password required on re-entry	Pass

Requirement	Action / Sub-Requirement	Expected Output	Actual Output	Pass/Fail
	Alert user on receipt of new logs	System tray icon flashes when there are new logs for viewing	System tray icon flashes when there are logs to view	Pass
	Display log information in tabular format	Display log in a table. Columns should allow sorting.	Logs displayed in tabular format. No sorting functionality.	Pass & Fail (*)
	Display chat transcripts in separate window	Chat transcript displayed in its own text area	Chat displayed in own window	Pass
	Allow log deletion from console	Delete selected log from list	Can delete selected log from list	Pass
	Allow logs to be exported into a text file	Exports selected log to a text file	Allows export of selected log to text file	Pass
Ruleset Configuration	Define SecureNet IP address	Provide a textbox in which to define the server IP	Textbox to define SecureNet server settings present	Pass
	Predefined categories holding relevant URLs	Provide a set of predefined categories with a combo box to allow configuration	Predefined categories present and working	Pass
	Provide ability to define custom allow / block lists	Provide a textbox in which to define allow / block lists. Lists will be shown within a text area.	Facility provided	Pass
	Provide ability to define banned words and phrases	Provide a textbox in which to define banned words and phrases. List will be shown within a text area.	Facility provided	Pass
	Provide ability to define sensitive personal information from being transmitted	Provide a textbox in which to define sensitive information. This list will be added to the set of violation keys	Sensitive information can be defined through users	Pass
	Set log options including criteria, upload frequency	Set logging options allowing the definition of criteria and upload frequency for each of web and chat	Logging options provided	Pass

Requirement	Action / Sub-Requirement	Expected Output	Actual Output	Pass/Fail
	Allow modification of existing custom lists	Ability to select existing information in custom-defined lists for redefinition or deletion	Facility provided	Pass
	Allow 'push' configuration update	Updates the configuration file on the remote client	Able to push configuration updates to clients	Pass (**)

Figure 7.2 Test Results: SecureNet Admin

(*) Not implemented due to time constraints.
(**) Has been observed to fail on rare occasions.

From the results of the tests, we can conclude that the system (with 2 minor exceptions) is able to perform as specified in our requirements. Ideally, we would also have liked to perform usability testing with some target users, but unfortunately, a combination of the magnitude of the implementation process, and inaccessibility to the ideal user groups, meant that there was insufficient time and resources to conduct a thorough usability test. However, from the experiments conducted using the software, we were extremely happy with the end results of the implementation since it successfully performed all of the required web filtering and chat monitoring flawlessly. Below are some screenshots demonstrating the chat censor function in operation:

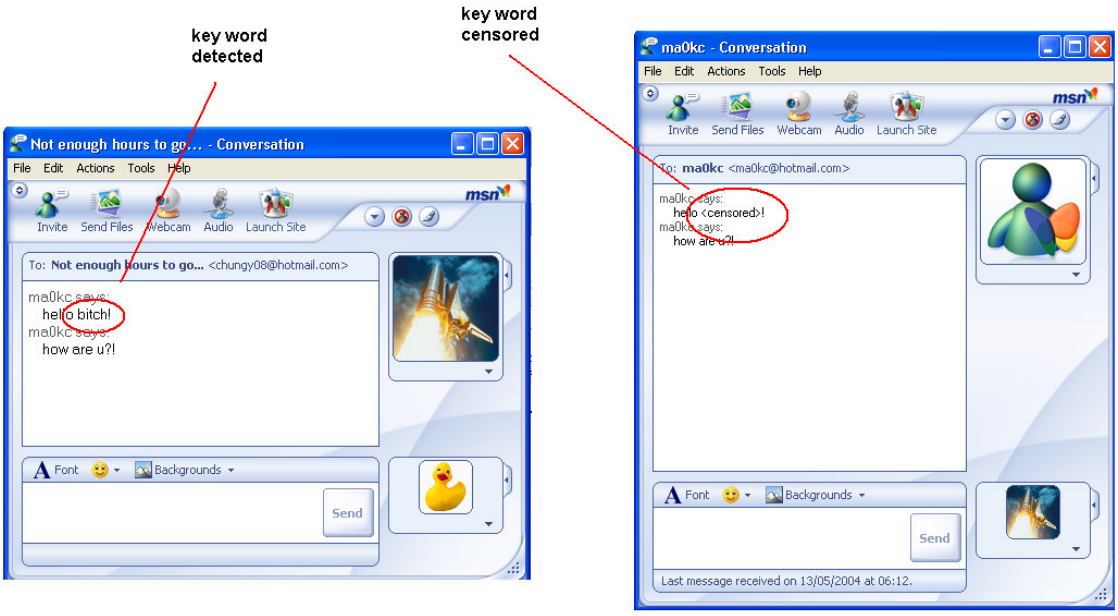

Figure 7.1 Censoring violating key words

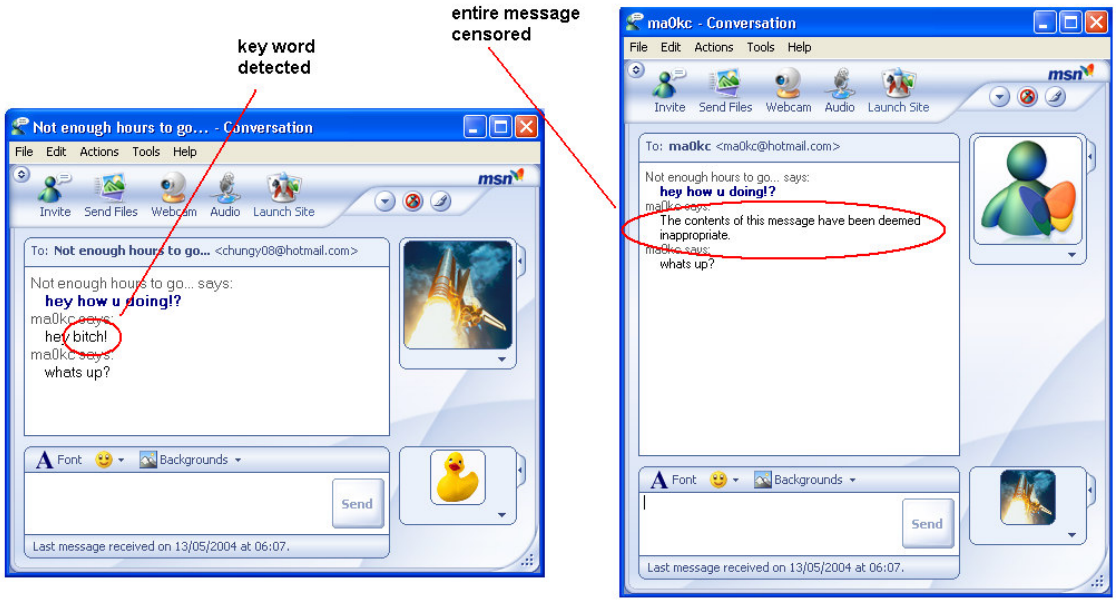

Figure 7.4 Censoring the entire message

There were three areas of interest that arose from testing:

1. As keywords are censored using regular expressions, it allows the censoring of all permutations of a key word. For example, the word 'bitch' is on the banned list, and as such it is censored, along with 'bitches', 'bitching' etc. This architecture goes 'wrong' when we have an ordinary word which also has the banned word within it. For example, if the word 'ass' were on the banned list, we would also be censoring it in normal words like 'pass', 'passing', 'glass' etc. Obviously this provides a problem, as a lot of ordinary words will have parts of them censored where the banned string is present within the word.

2. During testing, it was observed that on occasions, instances of logs and configuration files were not successfully transferred. Debugging the problem indicated that although we defined the size of the file before transmission, the value given by 'ret' to determine the number of bytes sent does not always equal the file size. As such the entire is not always sent during transmission. However, this occurred extremely randomly and it could not be ascertained as to why it was happening during the debugging process, and as such no fix was implemented. A consequence of this is that we have had to comment out the code that clears the log files after they are sent, so to ensure that the logs are not cleared before they have been successfully sent. A side issue from this is that we will get duplicates in the server logs unless we manually clear the logs on the client side after successful transmission.

3. Finally, the client-server architecture also allows the 'client' to be installed on the same computer as the 'server' as you can define the server address to be the IP address of the local computer (not the loopback address). As such, we have created an application that also operates as a stand-alone solution, thereby widening its potential market base.

8 Conclusion

8.1.1.1 Further work

Although the software developed offers much functionality and fulfils our requirements, it must be stressed that the package as a whole is still a prototype with a large number improvements to be made.

Primarily, the categories list would hugely benefit from extensive research allowing the definition of potentially millions of websites onto each list. The intent is to make these categories 'educational' rather than using them solely as blocklists (with the exception of the adult mature category), thereby allowing the use of the whitelist filters which permits access to only websites already known to be 'safe'.

Another area that needs more development is the implementation of the chat monitoring to a lot more chat applications. As we now have a base from which to develop, it would be relatively trivial to expand this onto the different chat applications by analysing the construction of their packets and then implementing a method to monitor chat applications of that type.

Referring back to requirements analysis, there were several areas identified which can be also developed, most notably the implementation of user groups that would allow different rulesets to be applied to different sets of users, and also remote desktop allowing an administrator to see what's going on on a client screen in real-time.

Additionally, several other implementational possibilities came to light during development. Firstly, referring back to chapter 6, there was an issue that the key word detector would also censor parts of 'ordinary' words that had within it the key word. As such, to prevent this from occurring, we could implement a dictionary object of ordinary words which would be excluded from censoring. Secondly, it is believed that the keyword detection method could also be applied to the website filtering part of the application, allowing it to censor websites from being viewed if it contained over the allowed threshold number of keywords. Additionally, the method could also be used to prevent the transmission of sensitive data in websites.

8.1.1.2 Concluding Remarks

In general, we were delighted with the end results of the project, given that the implementation successfully performed all of the web filtering and chat monitoring required of it after a substantial implementational effort completed in a limited period of time. The server-side software component provided an excellent graphical user interface with which an administrator could easily view violations detected by the analysing component, and also for the configuration of a significant number of rules.

The software component was developed after identifying a growing problem around the world of protecting children online, from the dangers of online predators and harmful online content after the very recent boom in numbers connected to the Internet since the introduction of unlimited Internet access through subscription services and Broadband.

The existing solutions arena was analysed, and a gap in the market was identified in specialising a product which provided website filtering and chat conversation monitoring in an effective way. Additionally, it was identified that a key area in the future market will be the use of remote administration and control applications which could remotely monitor and control a child's computer on a home or school local area network.

As such, the design and implementation stage had to consider and develop a client-server solution which offered the functionality required for a feasible solution, given the huge time and resource constraints of a final year dissertation. The solution designed employed the latest development technologies in Microsoft .Net, an ideal platform for building applications for the future, given that it allows the rapid creation and implementation of feature-rich Windows applications, exploiting many existing Windows technologies.

The final system achieved the initial requirements, and provides a base on which to add further functionality and improvements identified both in the original requirements analysis, but were overlooked for implementation due to time constraints, and also at improvements identified during the development process itself.

Bibliography

Baldwin, R. (Date of publication unknown). Survey of Spyware Tools and Countermeasures. [Online] http://www.plusfive.com/spywareV9slides.pdf (Accessed 6/12/03)

Bennett, S et al. (2001). Schaum's Outlines. UML. McGraw-Hill.

Carter, H. (2003). Microsoft Chat Rooms to Close After Abuse Fear. The Guardian, September 24. [Online] http://www.guardian.co.uk/online/news/0,12597,1048491,00.html (Accessed 5/12/03).
Switching Off… Chat Rooms Proving too Hot for Children to Handle. The Guardian, September 24. [Online] http://www.guardian.co.uk/microsoft/Story/0,2763,1048420,00.html (Accessed 5/12/03)

Chivers, I. (2003). Essential C# Fast. Springer.

Dix, A. et al. (1998). Human Computer Interaction, Second Edition. Prentice Hall.

Blum, R. (2003). C# Network Programming. Sybex.

Dixon, R et al. (2001). Chat Wise, Street Wise – Children and Internet Chat Services. [Online] http://www.internetcrimeforum.org.uk/chatwise_streetwise.pdf (Accessed 1/12/03)

Forouzan, B. (2003). TCP/IP Protocol Suite. McGraw-Hill.

Garfinkel, S. (1997). Web Security & Commerce. O'Reilly.

Glass, B. (2002). Activity Monitoring.
[Online] http://www.pcmag.com/article2/0,4149,53942,00.asp (Accessed 5/12/03)

Gould, P. (2002). Paedophiles Who Roam the Internet. BBC News Online.
[Online] http://news.bbc.co.uk/1/hi/uk/2492193.stm (Accessed 1/12/03)

Hirsh, L. (2001). The Boss Is Watching: Workplace Monitoring on the Rise.
[Online] http://www.newsfactor.com/perl/story/11634.html (Accessed 5/12/03)

Hughes, D. (2001). Kids Online: Protecting Your Children in Cyberspace
[Online] http://www.protectkids.com/dangers/ (Accessed 25/11/03)

Kennedy, K. (2001). Filters – What You Need to Know.
[Online] http://www.techlearning.com/db_area/archives/TL/200103/newsextra.html (Accessed 5/12/03)

Libery, J. (2003). Programming C#, 3rd Edition. O'Reilly.

Mega, M. and Syal, R. (2003). Global Force Of Internet Police To Guard Children From Paedophiles. The Daily Telegraph. [Online] http://www.telegraph.co.uk/news/main.jhtml?xml=%2Fnews%2F2003%2F11%2F30%2Fngroom30.xml&secureRefresh=true&_requestid=32166 (Accessed 5/12/03)

Mentalis. (2002). A Proxy Server in C#.
[Online]. http://www.mentalis.org/soft/projects/proxy/ (Accessed 1/5/04).

Munro, J. (2003). Protect Your Kids Online. PC Magazine.
[Online] http://www.pcmag.com/article2/0,1759,1135946,00.asp
(Accessed 5/12/03)

Petzold,C. (2002). Programming Windows With C#. Microsoft Press.
Pook, S. (2000). Internet Predator Jailed for 5 Years After Seducing Girl, 13.
The Daily Telegraph.
[Online] http://www.telegraph.co.uk/news/main.jhtml?xml=/news/2000/10/25/npaed25.xml
(Accessed 5/12/03)

Preece, J. et al. (2002). Interactive Design: Beyond Human-Computer Interface. Wiley.

Pressman, R. (1997). Software Engineering A Practitioner's Approach. Fourth Edition. McGraw-Hill.

Ropelato, J. (2003). Internet Filter Review.
[Online] http://www.internetfilterreview.com/index.html (Accessed 30/11/03)

Schmidt, H. (2003). C# A Beginner's Guide. Osbourne/McGraw-Hill.

Shoniregun, C. and Anderson, A. (2003). Is Child Internet Access a Questionable Risk?
[Online] http://www.acm.org/ubiquity/views/v4i29_shoniregun.html (Accessed 4/12/03)

Sommerville, I. (2001). Software Engineering, 6th Edition. Addison Wesley.

Steele, J. (2001). 130 Arrested Over Internet Child Porn. The Daily Telegraph.
[Online]
http://www.telegraph.co.uk/news/main.jhtml?xml=%2Fnews%2F2001%2F11%2F29%2Fnpaed29.xml (Accessed 5/12/03)

Stevens,R. (1994). TCP/IP Illustrated, Volume 1 The Protocols. Addison-Wesley

Stiller, E, LeBlanc, C. (2001). Project Based Software Engineering. An Object-Oriented Approach. Addison Wesley.

Thornburgh, D. et al. (2002).
Youth, Pornography, and the Internet. Publisher: National Academy Press.
[Online] http://www.nap.edu/books/0309082749/html/ (Online version) (Accessed 6/12/03)

URL -BCS. British Computer Society. (2003). Children and the Internet.
[Online] http://www1.bcs.org.uk/ (Accessed 29/11/03)

URL-DK. Dot Kids Domains. (2003). [Online] http://www.kidsdomains.org
(Accessed 8/12/03)

URL-FBI. Federal Bureau of Investigation (US). (Date of publication unknown).
FBI Publications – A Parent's Guide to Internet Safety.
[Online] http://www.fbi.gov/publications/pguide/pguidee.htm (Accessed 25/11/03)

URL -ICRA. Internet Content Rating Association. (2003).
[Online] http://www.icra.org/ (Accessed 29/1/03)

URL -IWF. Internet Watch Foundation. (2003).
[Online] http://www.iwf.org.uk/safe_surfing/kids_tips.html (Accessed 29/11/03)

URL -LAN. Cyberspace Research Institute, Lancaster University. (2002). Young Peoples Use of Chat Rooms: Implications for Policy Strategies and Programs of Education.
[Online] http://www.uclan.ac.uk/facs/science/psychol/Homeoff2.pdf (Accessed 2/12/03)

URL-POL. Department for Education and Skills. (2003). Parents Online.
[Online] http://www.parentsonline.gov.uk/parents/index.html (Accessed 26/11/03)

URL –NUA. NUA Internet Surveys. (2003). How Many Online?
[Online] http://www.nua.ie/surveys/how_many_online/europe.html (Accessed 26/11/03)

URL-RSAC. Recreational Software Advisory Council. (1999).
[Online] http://www.rsac.org/ (Accessed 6/12/03)

URL-SOB. The Home Office. (2000). Sexual Offences Bill.
[Online] http://www.homeoffice.gov.uk/justice/sentencing/sexualoffencesbill/index.html
(Accessed 5/12/03)
Summary of Changes. [Online] http://www.homeoffice.gov.uk/docs/set_summ.pdf
(Accessed 5/12/03)

URL-SPAM. The Spamhaus Project. (2003). Spamhaus Block List.
[Online] http://www.spamhaus.org/sbl/ (Accessed 6/12/03)

URL-WTN. The Home Office. Wise Up to the Net. (2003). Keeping Your Child Safe on the Internet. [Online] http://www.homeoffice.gov.uk/docs/childsafetyinternet.pdf
(Accessed 5/12/03)

Quinlan, F. (2001). Parent's Urged to Monitor Children's Internet Surfing.
The Irish Examiner. [Online] http://archives.tcm.ie/irishexaminer/2001/07/18/story8226.asp
(Accessed 25/11/03)

Ward, D. (2003). Ex-US Marine Flown to Britain. The Guardian, August 22.
[Online] http://www.guardian.co.uk/uk_news/story/0,3604,1027249,00.html
(Accessed 5/12/03)

Willard, N. (2002). The Use of Filtered Monitoring in the Context of a Comprehensive Strategy to Address the Safe and Responsible Use of the Internet by Students.
[Online] http://www.vericept.com/download/whitepaper/vericept_education.pdf
(Accessed 6/12/03)

Zitz, M. (2003). Internet Safety 101.
[Online] http://www.parentsonline.gov.uk/2003/parents/links/Online_safety_parents.html
(Accessed 5/12/03)

Appendix A: Use Case Analysis

A.1 Logon Use Case Description

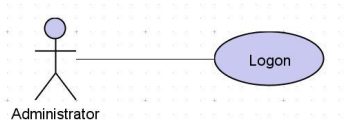

Figure A.1.1 Logon Use Case

Name	Logon		
Brief Description	Logs an administrator onto the administration console. The console allows definition and configuration of client-system rulesets, the viewing of log files and alerts, the granting of access to blocked URLs upon request, and remote desktop facility.		
Post-conditions	Provides assess to administration console with correct level of privilege.		
Flow of Events		**Actor input**	**System response**
	0	Opens administration console	
	1		Requests username and password
	2	Provide username and password	
	3		Grants access to administration console
	4	Perform action(s)	
	5		Carries out action(s)
	6	Logout	

Table A.1.1 Logon Use Case Description

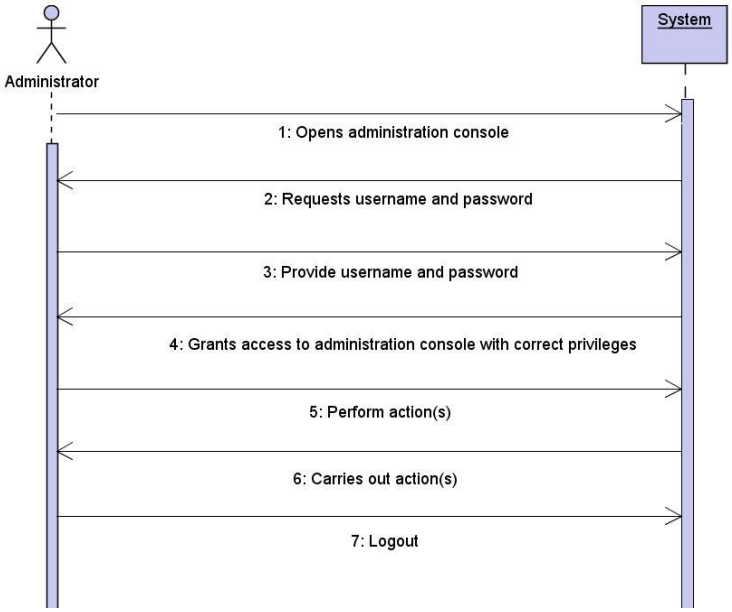

Figure A.1.2 Logon Sequence

A.2 Configure Ruleset Use Case Description

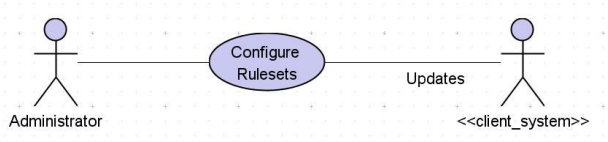

Figure A.2.1 Configure Ruleset Use Case

Name	Configure Ruleset
Brief Description	Allows an administrator to define the rules to which a client_system adheres. Definable rules include: 1. Blocked URLs 2. Allowed URLs 3. Logging of URLs visited 4. Logging of chat conversations 5. Alert options 6. 'Bad' word filters 7. Sensitive information filters System will update the ruleset of client_system after administrator saves ruleset configuration and apply's the ruleset.
Pre-conditions	Administrator must be logged on to administration console with sufficient privileges.
Post-conditions	Updates rulesets and pushes new rules onto client_system.

Flow of Events

	Actor input	System response
0	Select configure ruleset option	
1		Provides ruleset configuration options dialog
2	Edit required fields and submits configuration	
3		Saves configuration options and displays updated settings
4		Pushes updates configuration to client_system

Table A.2.1 Configure Ruleset Usecase Description

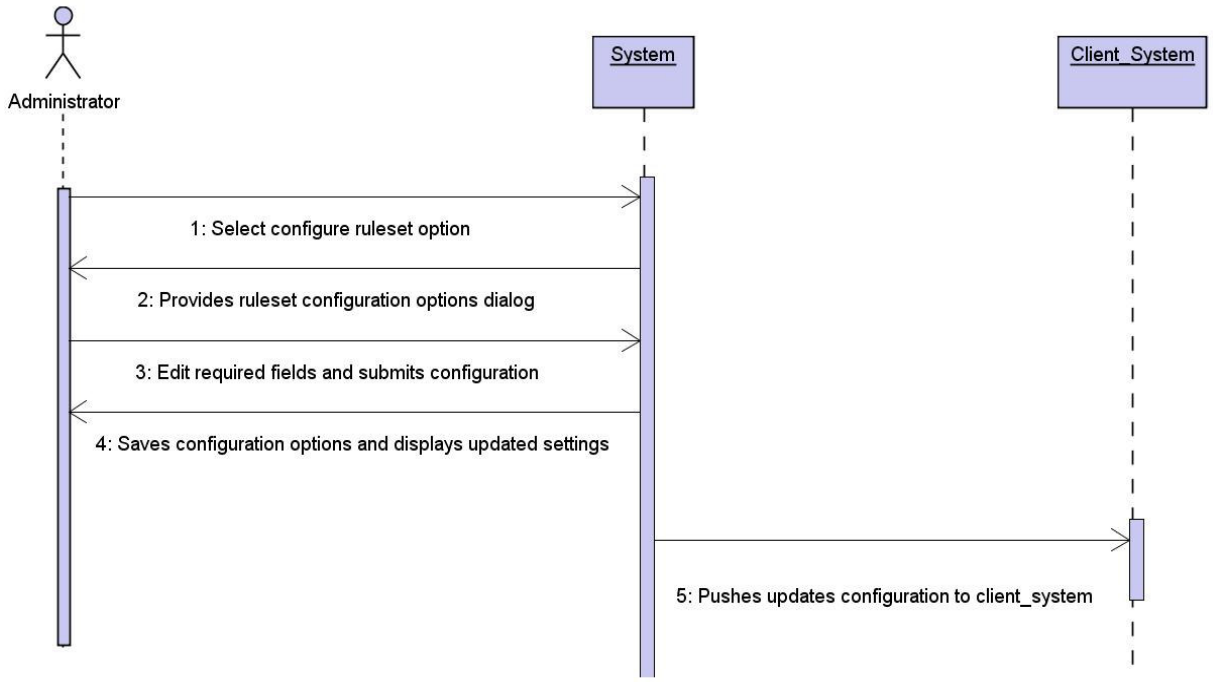

Figure A.2.2 Configure Ruleset Sequence

A.3 View Logs Use Case Description

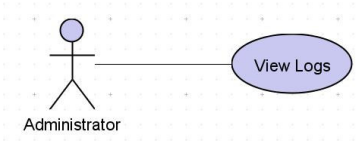

Figure A.3.1 View Logs Use Case

Name	View Logs
Brief Description	Allows an administrator to view logs of: 1. Attempted access to blocked URLs 2. View logs of chat conversation where inappropriate words or conversation were detected Chat logs should contain a transcript of the entire conversation, allowing the administrator to understand the context of the violation. Administrator shall be able to order logs by: 1. Severity of offence 2. User 3. Date and time

Pre-conditions	Administrator must be logged on to administration console with sufficient privileges.		
Flow of Events		**Actor input**	**System response**
	0	Selects view web logs option	
	1		Displays web log dialog
	2	Selects attempted access to blocked URLs log	
	3		Displays a list of URLs which the user attempted to access and the date and time of offence
	4	Selects to order log by severity, user or time	
	5		Orders the log by severity, user or time (depending on option selected by administrator)
	6	Selects filter log	
	7		Displays a filter log dialog
	8	Enters the parameters of the filter	
	9		Displays a list of URLs which the user attempted to access within the constraints of the defined filter
	10	Selects chat conversations log	
	11		Displays a text transcript of a chat conversation where inappropriate content was detected
	12	Selects to order log by severity, user or time	
	13		Orders the log by severity, user or time (depending on option selected by administrator)
	14	Selects filter log	
	15		Displays a filter log dialog
	16	Enters the parameters of the filter	
	17		Displays a text transcript of chat conversations within the constraints of the defined filter

Table A.3.1 View Logs Use Case Description

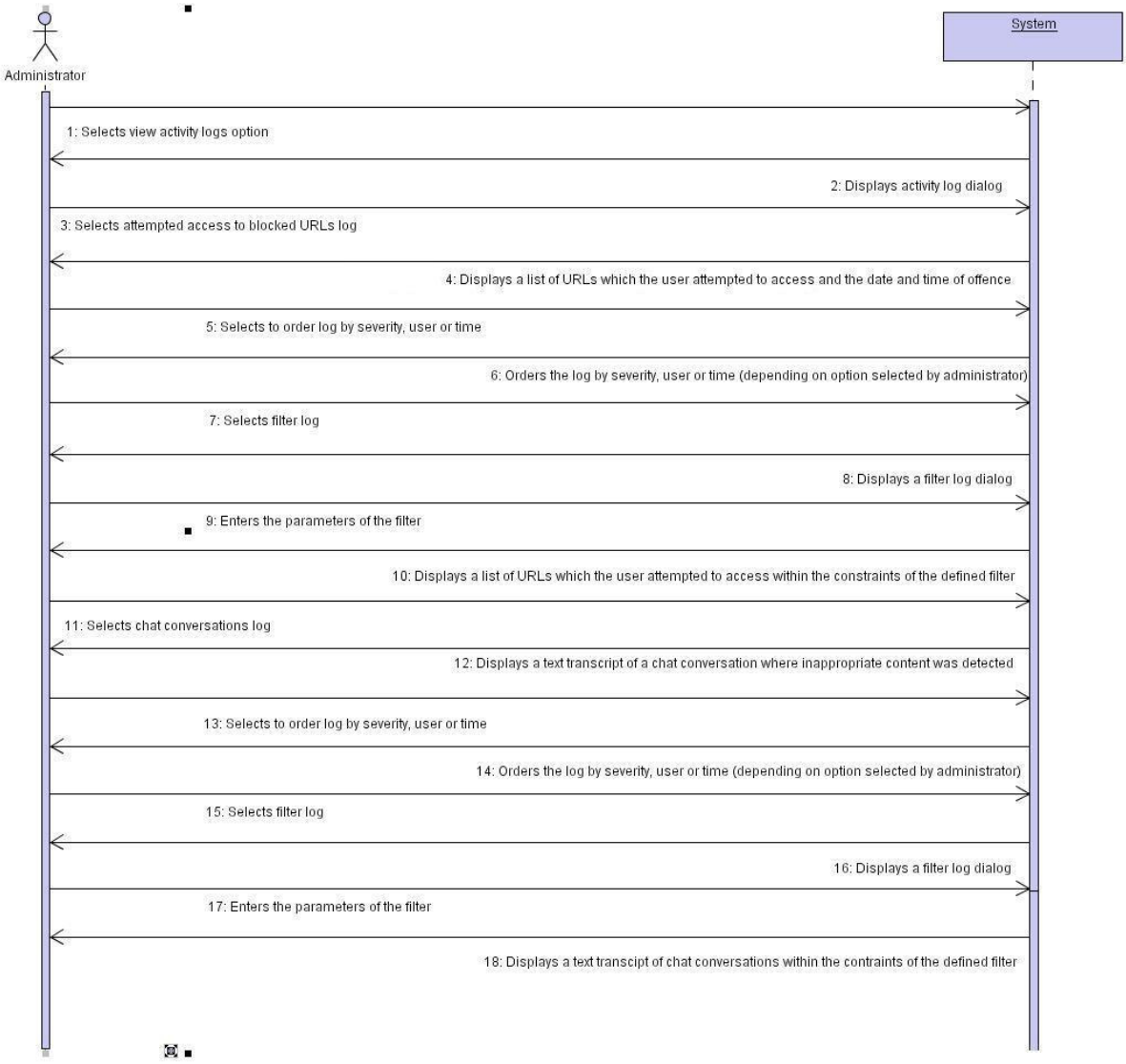

Figure A.3.2 View Logs Sequence

A.4 Configure User Groups Use Case Description

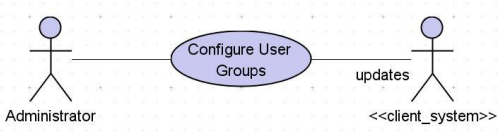

Figure A.4.1 Configure Groups Use Case

Name	Configure User Groups		
Brief Description	An administrator can create new user groups to assign a ruleset which differs from the default ruleset. User groups allow different ruleset configurations to be applied to different users. Administrators will be able to move users between groups.		
Pre-conditions	Administrator must be logged on to administration console with sufficient privileges.		
Post-conditions	Selected users in new group with differing ruleset configuration. Updates pushed to client_system.		
Flow of Events		**Actor input**	**System response**
	0	Selects user groups option	
	1		Displays user groups dialog
	2	Selects create group option	
	3		Requests name for new group
	4	Enters a name	
	5		Displays a list of users
	6	Selects users to add to group	
	7		Displays users in new group
	8		Display ruleset configuration dialog
	9	Configures ruleset for the group	
	10		Updates information to client_system

Table A.4.1 Configure Groups Description

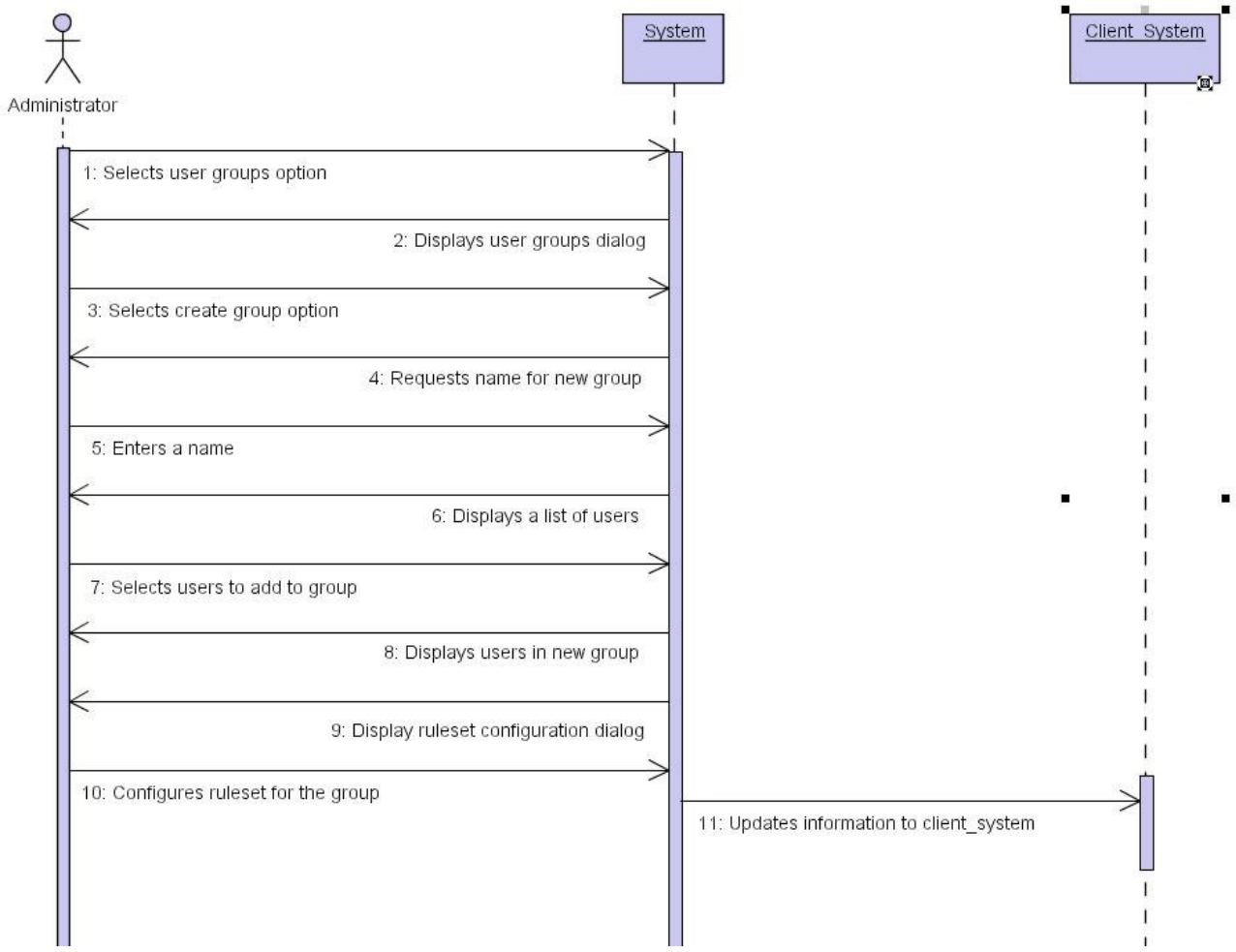

Figure A.4.2 Configure Groups Sequence

A.5 View Remote Desktop Use Case Description

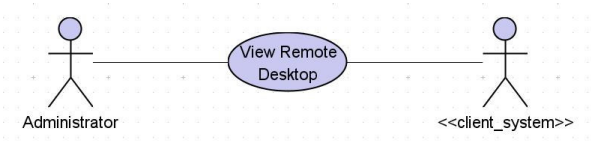

Figure A.5.1 View Remote Desktop Use case

Name	View remote desktop		
Brief Description	The client_system identifies inappropriate material in a conversation or severe infringement of acceptable internet use and alerts the administrator. If the log data is not sufficient and/or a log is not yet be available as the conversation has not finished, an administrator can view the conversation 'live' on the remote desktop for more information. If necessary, the administrator will be able to take control of the remote system.		
Pre-conditions	Administrator must be logged on to administration console with sufficient privileges.		
Flow of Events		**Actor input**	**System response**
	0		Alerts administrator of a severe infringement of acceptable chat or internet activity
	1	Selects view logs	
	2		Displays log information
	3	Decides there is insufficient information (no input)	
	4	Selects remote desktop option	
	5		Displays connect to remote desktop dialog
	6	Selects machine (with information of present user) to connect to	
	7		Connects to remote machine
	8		Displays remote desktop
	9	Selects control remote desktop option	
	10		Changes settings to allow remote-control
	11		Displays remote desktop with admin-control
	12	Perform action(s)	
	13	Selects disconnect remote desktop option	
	14		Disconnects from remote machine

Table A.5.1 View Remote Desktop Description

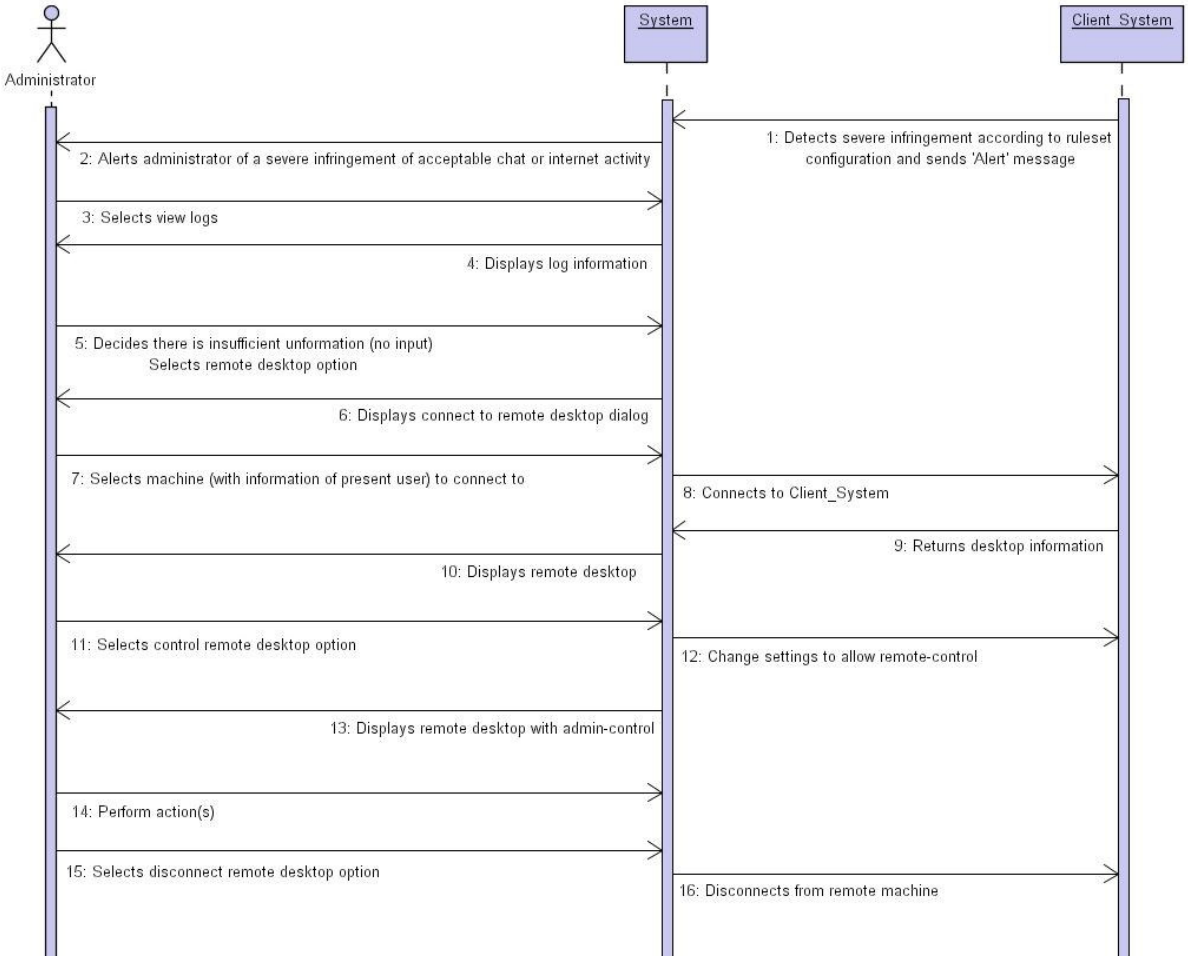

Figure A.5.2 View Remote Desktop Sequence

A.6 Permit URL Access Use Case Description

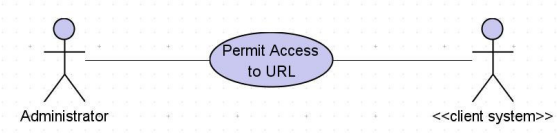

Figure A.6.1 Permit URL Access Use case

Name	Permit Access
Brief Description	A user may be blocked from accessing a legitimate URL by the client_system, or may have a legitimate reason to access a blocked URL. If this scenario occurs, a user shall be able to request permission to access the website in question. The administrator will receive the request and decide whether or not to permit access. The administrator shall be able to perform the following given a permit access request: 1. Deny the request 2. Permit the request to the requesting user 3. Permit & Add the URL to the 'global' permitted URLs list
Pre-conditions	Administrator must be logged on to administration console with sufficient privileges.
Post-conditions	A user will be informed that a URL request has been denied or accepted. If the administrator decides that the URL is incorrectly blocked, the system will automatically update the ruleset configuration by adding the URL to the permitted URL list.

Flow of Events

	Actor input	System response
0		Provides a request dialog from a user wanting access to a blocked URL
1	Deny request to user	
2		Sends 'Deny' message to client_system which informs user of denied request
3	Permit user request	
5		Sends 'Allowed' message to client_system which informs the user that the request has been granted
4		Sends 'Add URL' message to client_system which updates the permitted URL to allow access
6	Permit user request & Add URL to global Permitted URLs list	
7		Add URL to permitted URLs ruleset configuration
8		Update client_system ruleset
9		Send 'Allowed' message to client_system

Table A.6.1 Permit URL Access Description

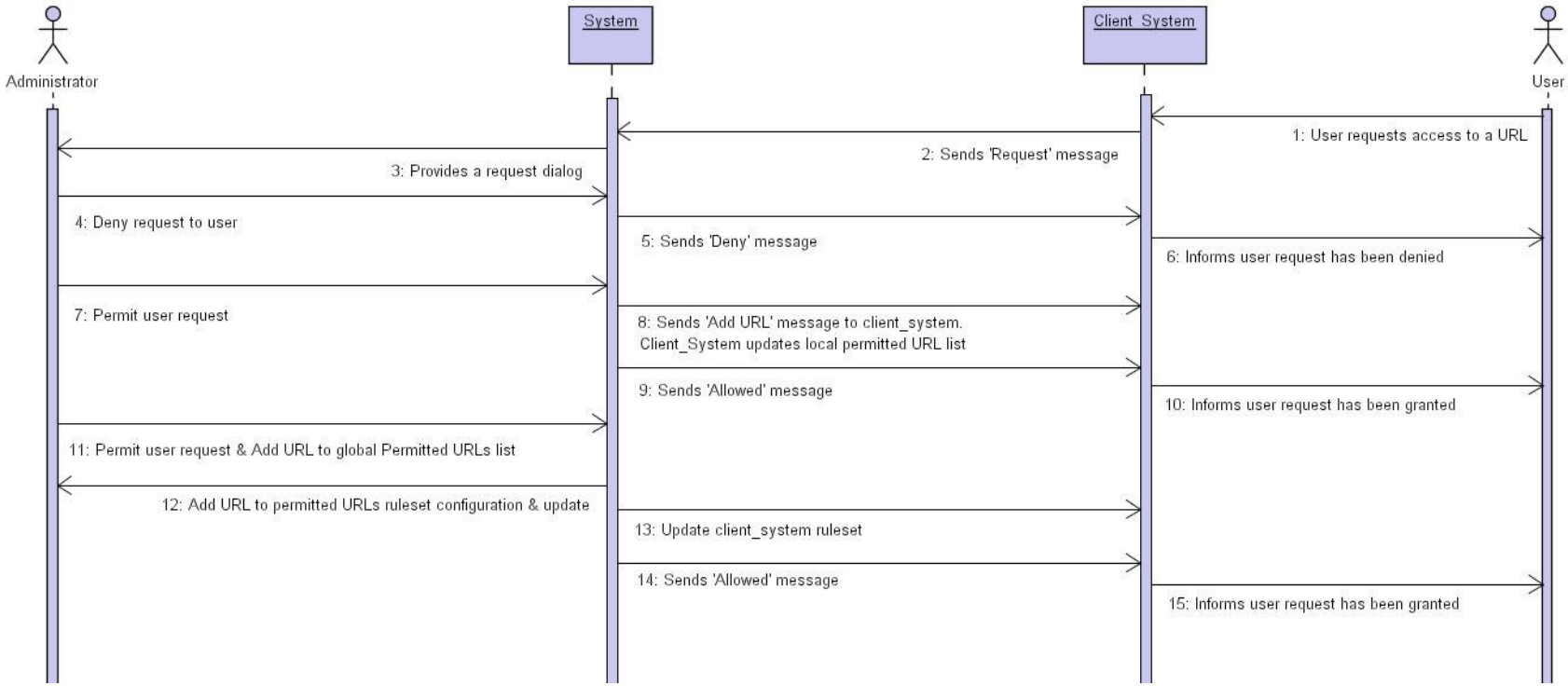

Figure A.6.2 Permit URL Access Sequence

A.7 Block URL Use Case Description

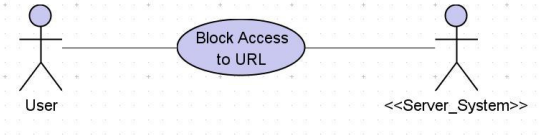

Figure A.7.1 Block URL Use Case

Name	Block Access to URL		
Brief Description	System detects when a user attempts to access a blocked URL and prevents access. The attempt is logged and sent to the server.		
Pre-conditions	User is browsing the internet		
Flow of Events		**Actor input**	**System response**
	0	Attempts to access a URL	
	1		Detects URL is on blocked list, denies access and logs the attempt
	2		Notifies user URL is prohibited
	3		Provide user with email address to request access to URL

Table A.7.1 Block URL Description

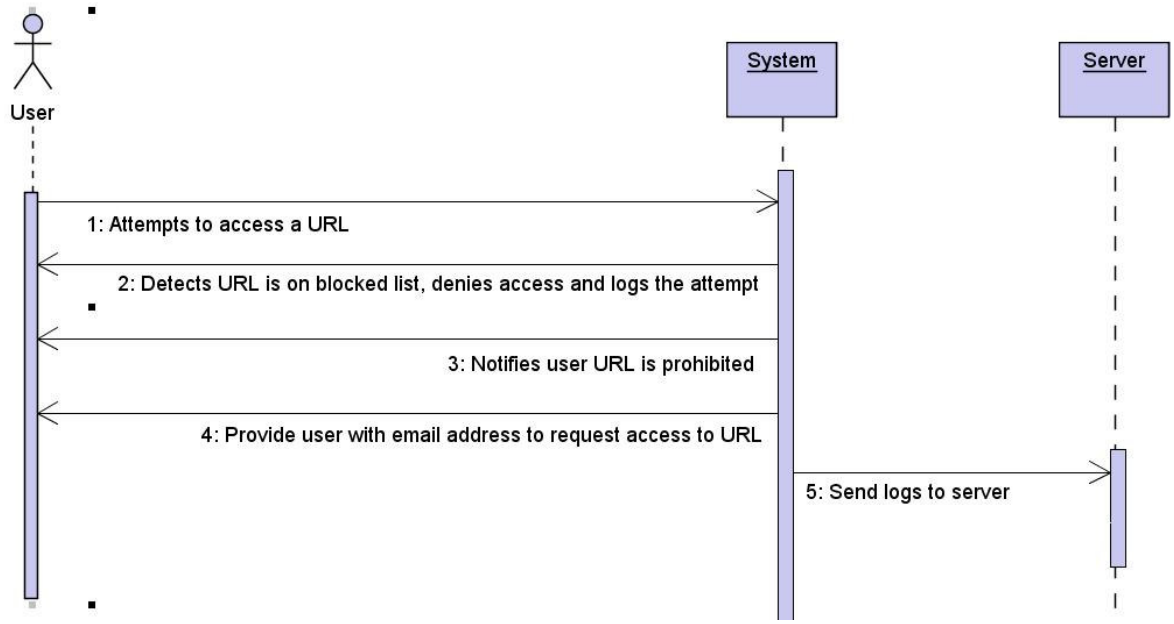

Figure A.7.2 Block URL Sequence

A.8 Block Inappropriate Chat Use Case Description

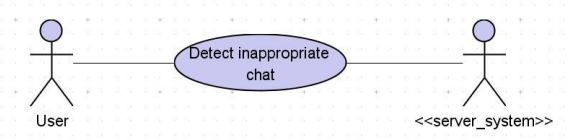

Figure A.8.1 Block Inappropriate Chat Use Case

Name	Detect Inappropriate Chat		
Brief Description	If pre-defined keywords are present in a chat conversation, the chat conversation is logged and a text transcript sent to the server. Pre-defined severity ratings determine whether an administrator should be alerted.		
Pre-conditions	A user is using an online chat program.		
Flow of Events		Actor input	System response
	0	Uses an online chat program	
	1		Detects inappropriate content in conversation
	2		Logs a transcript of the conversation
	3		Alerts the administrator depending on severity
	4		Identifies username and IP address of 3rd party
	5		Sends log to server

Table A.8.1 Block Inappropriate Chat Description

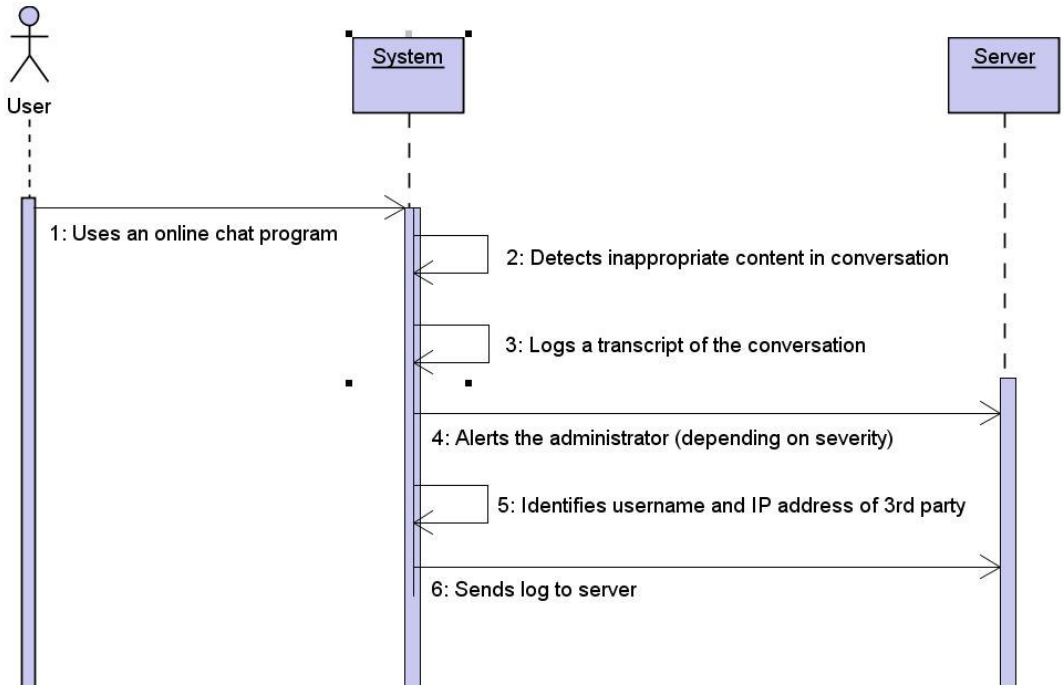

Figure A.8.2 Block Inappropriate Chat Sequence

Appendix B: Packet Analysis of MSN Messenger

B.1 MSN Messenger Sign-On Sequence

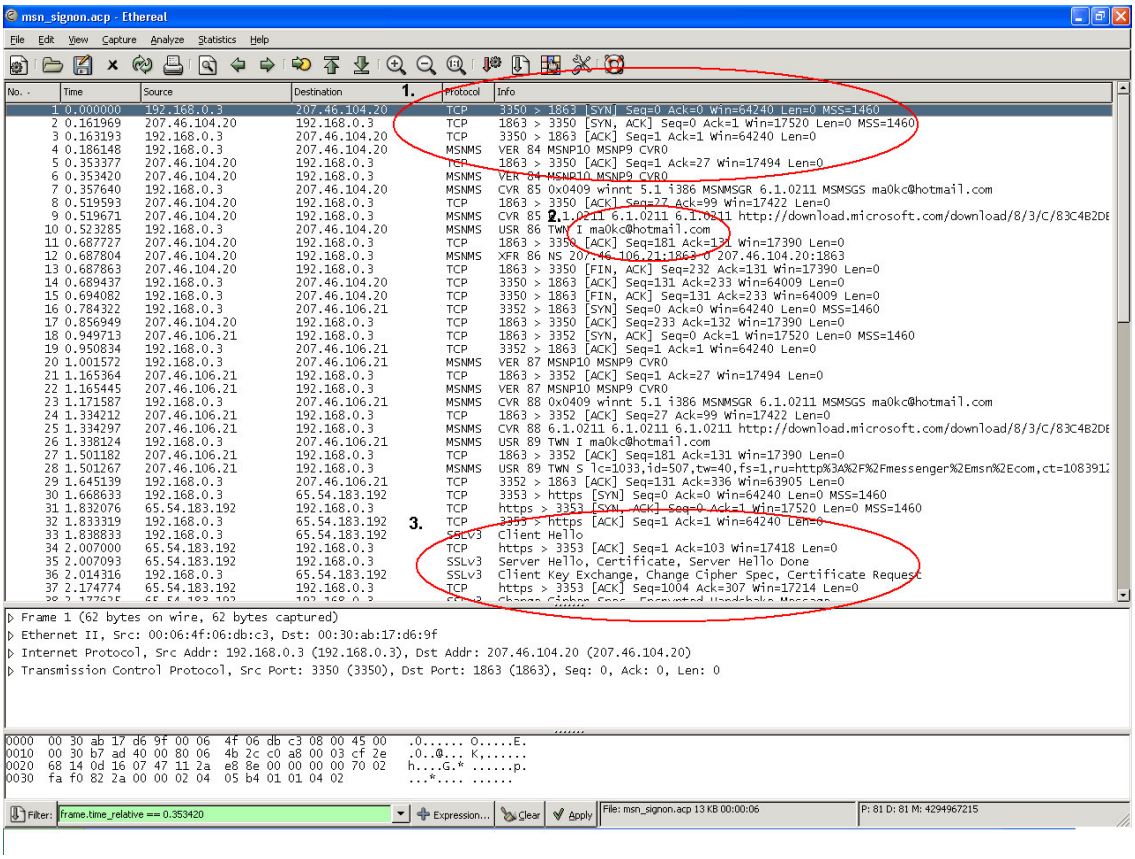

Figure B.1.1 MSN Messenger Sign On

On signing on to MSN Messenger, the procedure is as follows:

1. A 3-way handshake to establish a connection with a MSN server, followed by information detailing the version of the application that is attempting to connect.
2. The local user's email address is sent to the server
3. The user is authenticated with the server using SSL.
4. The user is authenticated. Open the user's contact book.
5. Retrieve list of contacts.
6. Check for new messages in Hotmail inbox.

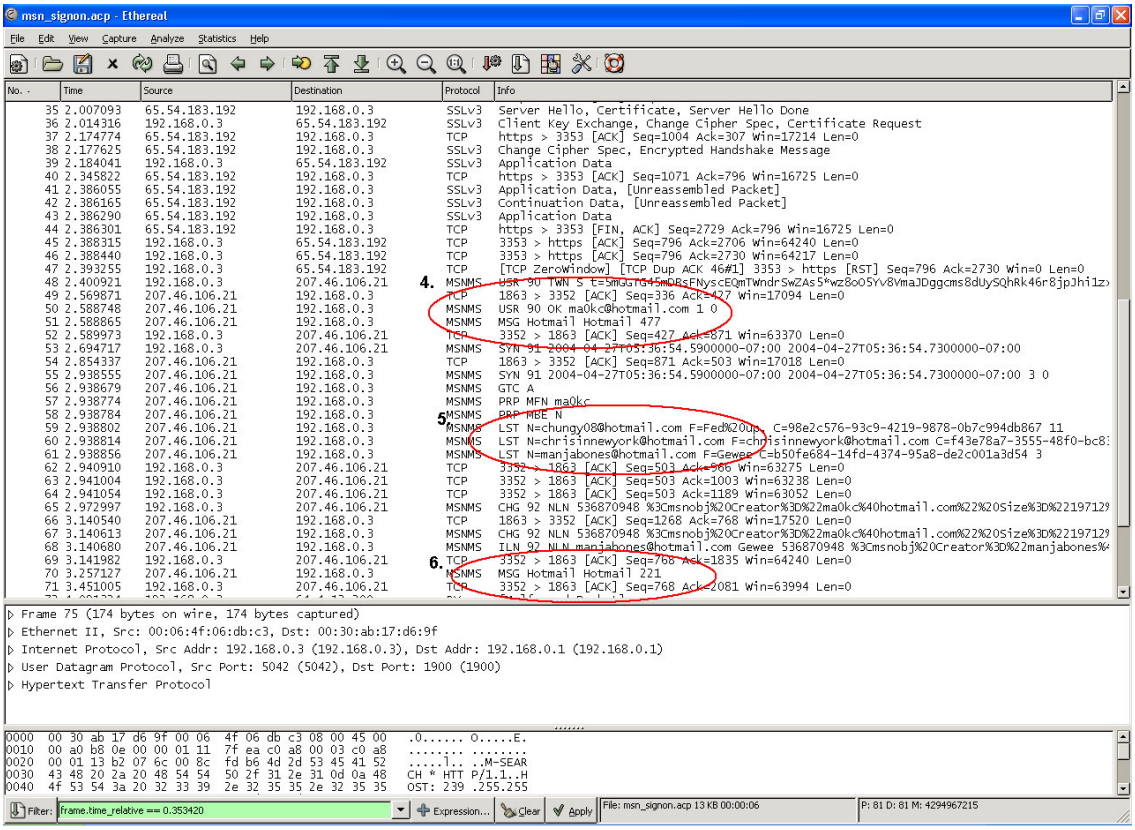

Figure B.1.2 MSN Messenger Sign On Continued

B.2 MSN Messenger Chat Sequence Initialisation

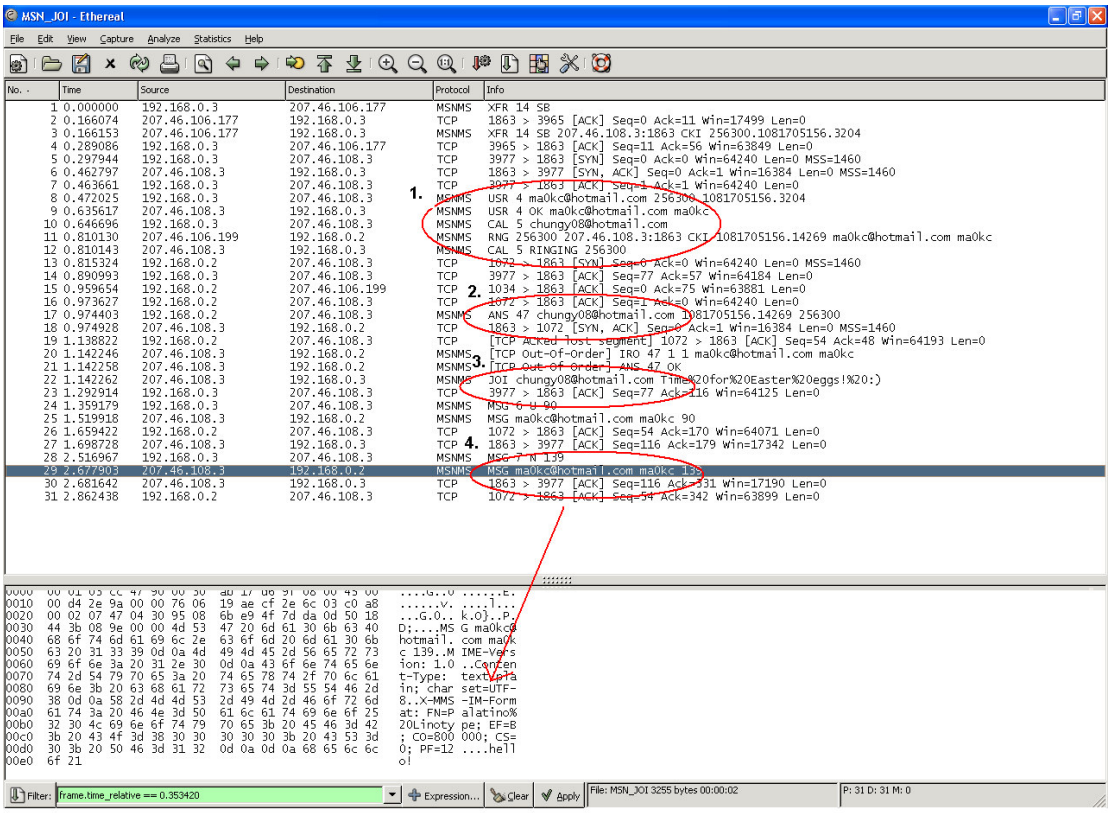

Figure B.2.1 Chat Initialisation Sequence, Local User

It is possible for either the local user, or a contact of the local user to initialise a chat conversation.

- The sequence for a local user initialising a conversation is as follows:

    1. The local user (ma0kc@hotmail.com) commences the chat initialisation sequence. The 'CAL' message indicates that the local user is attempting to contact someone on the users contact list. This is followed by the 'RNG' message, indicating that we are attempting to connect with and establish a conversation with that contact.

    2. The 'ANS' message indicates that the contact has accepted the local users connection request.

    3. This is followed by the 'JOI' message indicating that the user has joined the conversation.

    4. A 'MSG' message indicates that a message has been sent in the conversation. Closer inspection on the 'MSG' packet shows the user has sent the contact the message 'hello!'.

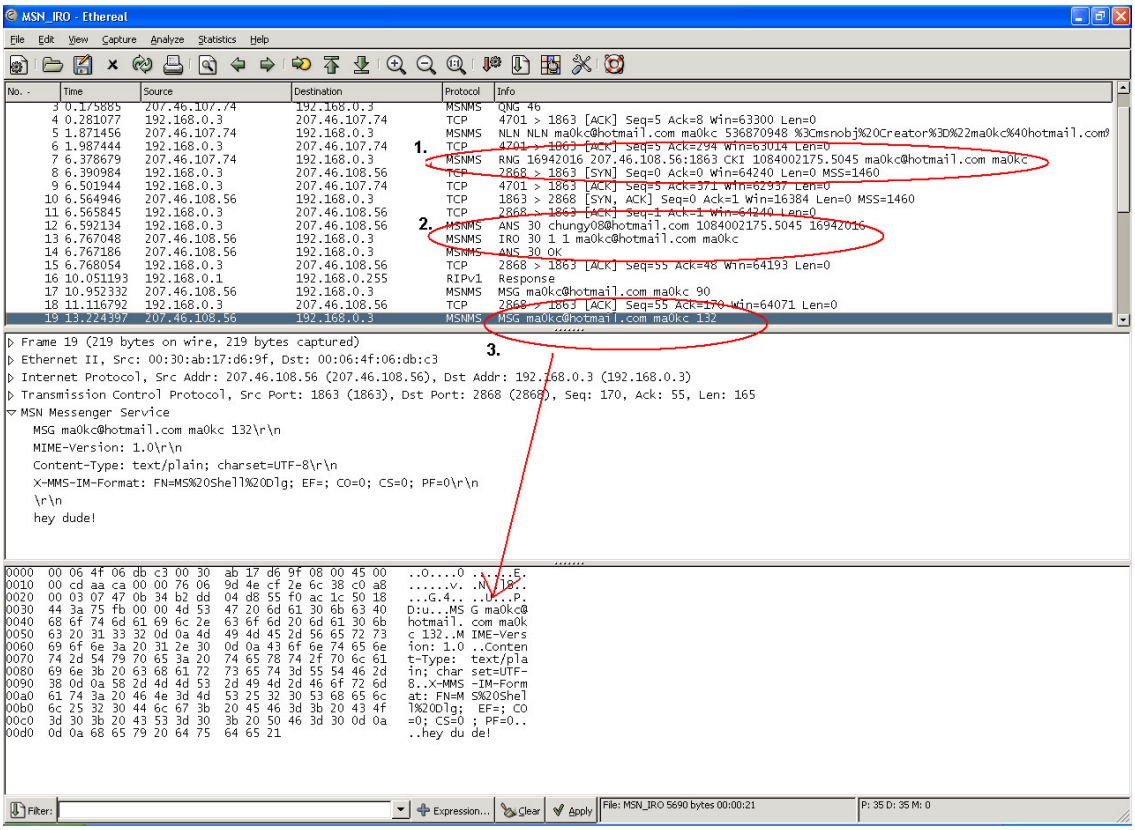

Figure B.2.2 Chat Initialisation Sequence, Contact User

- The sequence for a contact initialising a conversation is as follows:

    1. The contact user (ma0kc@hotmail.com) commences the chat initialisation sequence. The 'RNG' message indicates that the contact user is attempting to contact the local user to establish a chat conversation.

    2. The 'ANS' message indicates that the local user has accepted the contact users' connection request. This is followed by the 'IRO' message indicating that the local user has joined the conversation.

    3. A 'MSG' message indicates that a message has been sent in the conversation. Closer inspection on the 'MSG' packet shows the user has sent the contact the message 'hey dude!'.

From the analysis of Figures B.2.1, and B.2.2, it is subsequently possible to determine:

- A 'JOI' or a 'IRO' message must be established (indicating that both users have agreed to join a conversation) before any parsing of chat messages is required.

- A 'MSG' message indicates a chat conversation packet that needs parsing to extract the conversation contents.

B.3 MSN Messenger Chat Sequence

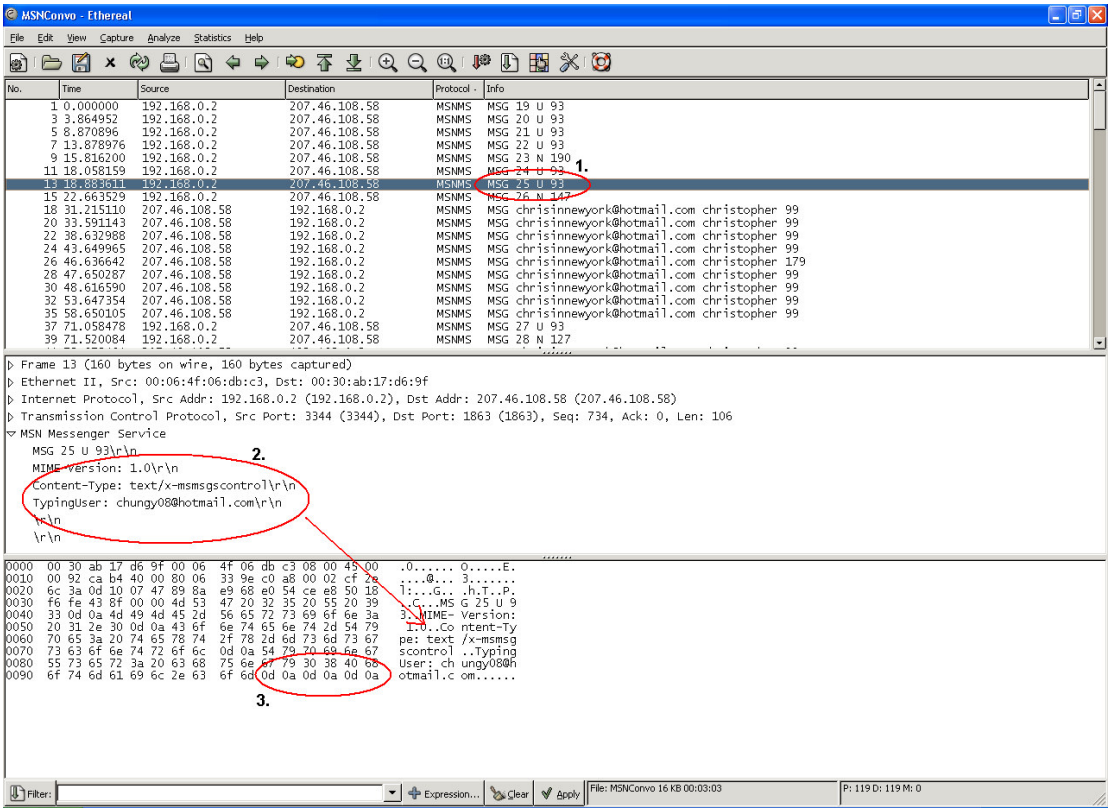

Figure B.3.1 Chat Sequence, Local User Typing

The chat conversation sequence can be broken down into several sections:

- Figure B.3.1 shows a MSN packet sent to the contact from the local user indicating that the user is currently typing a message. The packet structure is as follows:

    1. The packet header is observed to follow the format:

        MSG 25 U 93

    2. The packet contents are observed to be:

        a) MIME-Version: 1.0 (+ newline)
        b) Content-Type: text/msmsgscontrol (+ newline)
        c) TypingUser: chung08@hotmail.com (+ newline)

    3. The packet terminates with a sequence of 3 newlines.

NB. A newline is defined as the sequence carriage return followed by a line feed, and is represented as '0d 0a' in hexadecimal, or '\r\n' in C# code.

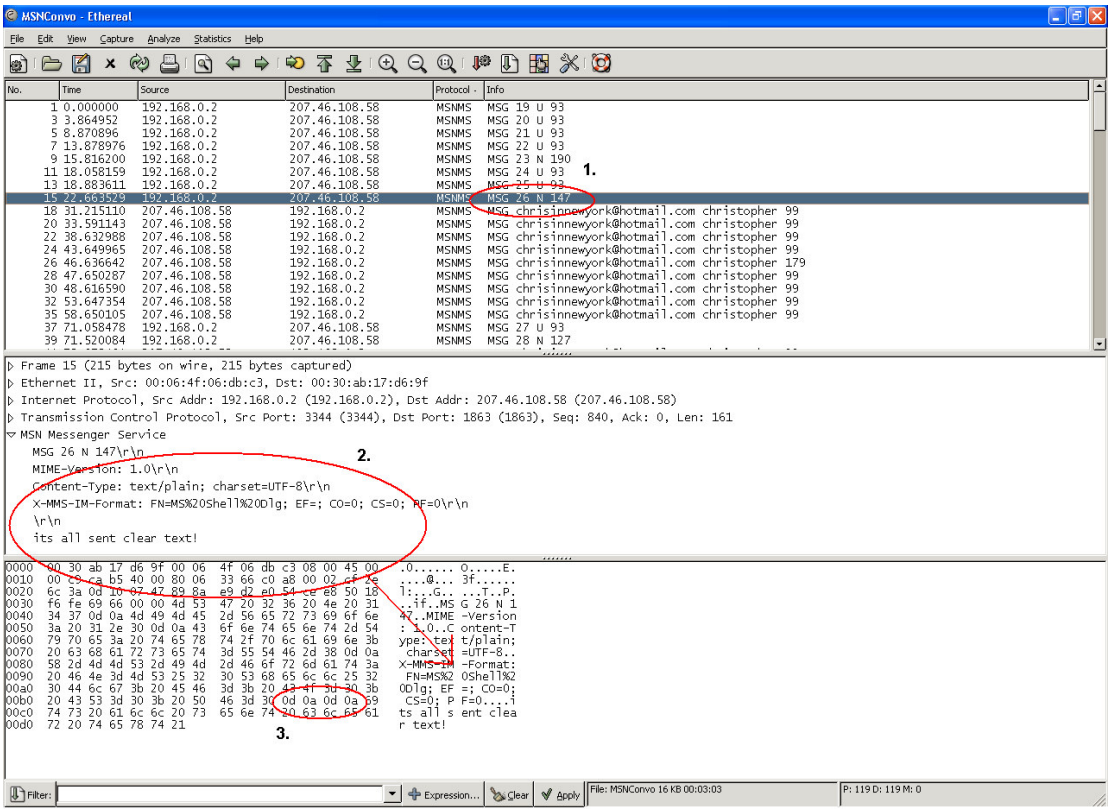

Figure B.3.2 Chat Sequence, Local User Send Message

- In continuation of Figure B.3.1, Figure B.3.2 shows a MSN packet after the local user has sent the message to the contact. The packet structure is as follows:

    1. The packet header is observed to follow the format:

        MSG 26 N 147

    2. The packet contents are observed to be:

        a) MIME-Version: 1.0 (+ newline)
        b) Content-Type: text/plain; charset=UTF-8 (+ newline)
        c) X-MMS-IM-Format: FN=MS%20Shell%20Dlg; EF=; CO=0; CS=0; PF=0 (+ newline)

    3. The chat message follows a sequence of 2 newlines:

        'its all sent clear text!'

From the analysis of Figures B.3.1, and B.3.2, it is subsequently possible to determine:

- In the packet header, a 'U' signifies that the local user is typing, where as a 'N' signifies the transmission of a message.

- In the packet contents, fields b) and c) change, depending on whether the local user is typing or sending a message.

- If a user is typing a message, we get a packet with an ending sequence of 3 newlines, where as in a message packet, we get a sequence of 2 newlines, followed by the chat message.

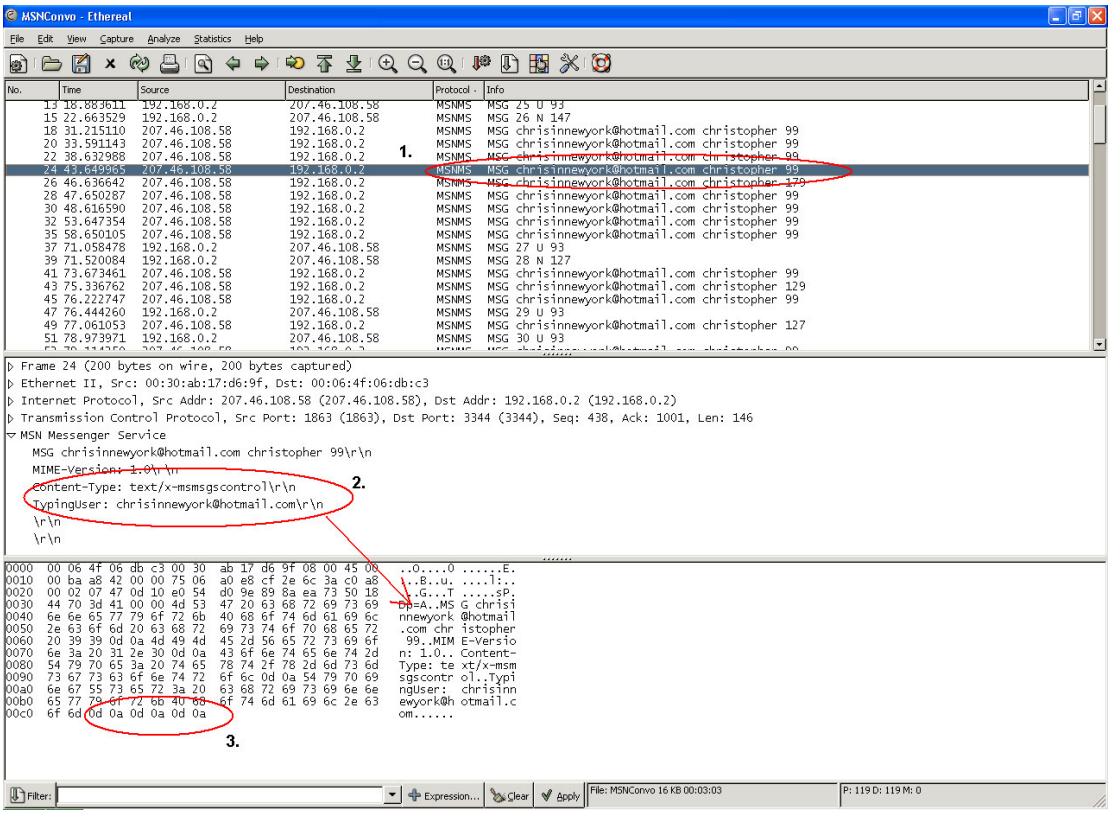

Figure B.3.3 Chat Sequence, Contact User Typing

- Figure B.3.3 shows a MSN packet sent to the local user from the contact indicating that the contact is typing a message. The packet structure is as follows:

    1. The packet header is observed to follow the format:

        MSG chrisinnewyork@hotmail.com christopher 99

    2. The packet contents are observed to be:

        a) MIME-Version: 1.0 (+ newline)
        b) Content-Type: text/msmsgscontrol (+ newline)
        c) TypingUser: chrisinnewyork@hotmail.com (+ newline)

    3. The packet terminates with a sequence of 3 newlines.

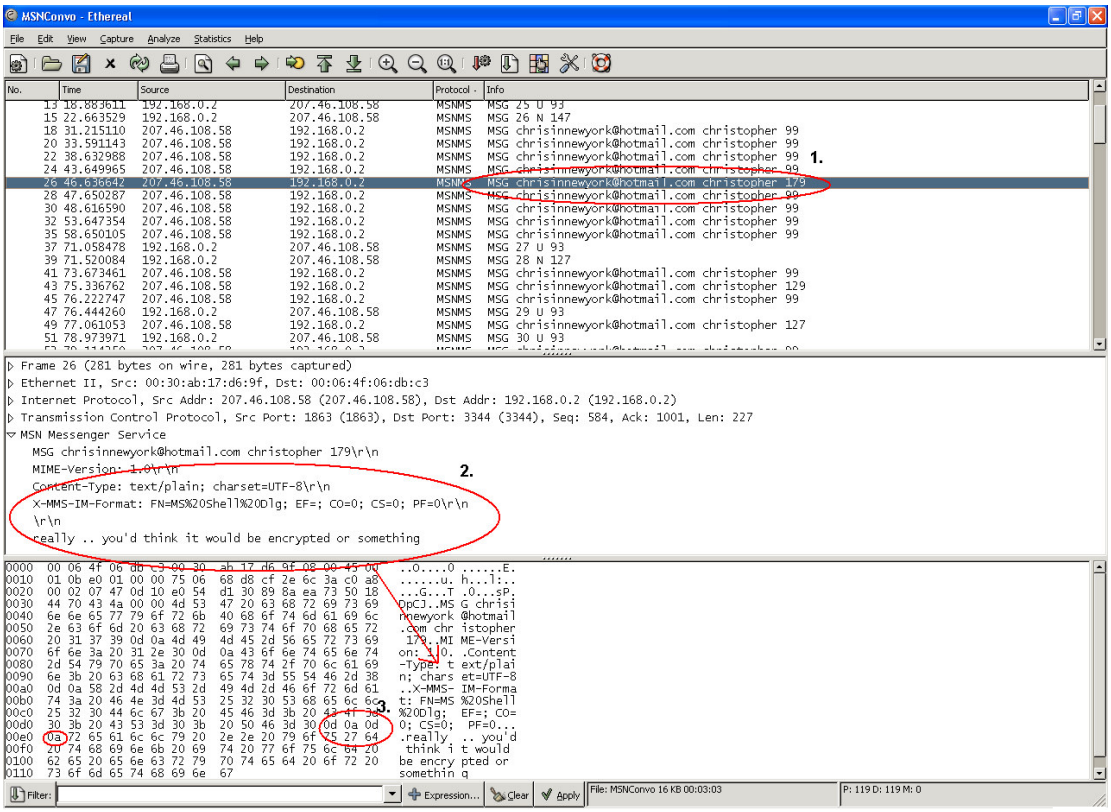

Figure B.3.4 Chat Sequence, Contact User Send Message

- In continuation of Figure B.3.3, Figure B.3.4 shows a MSN packet after the contact user has sent the message to the local user. The packet structure is as follows:

    1. The packet header is observed to follow the format:

        MSG chrisinnewyork@hotmail.com christopher 99

    2. The packet contents are observed to be:

        a) MIME-Version: 1.0 (+ newline)
        b) Content-Type: text/plain; charset=UTF-8 (+ newline)
        c) X-MMS-IM-Format: FN=MS%20Shell%20Dlg; EF=; CO=0; CS=0; PF=0 (+ newline)

    3. The chat message follows a sequence of 2 newlines:

        'really.. you'd think it would be encrypted or something'

We can now analyse Figures B.3.3 and B.3.4:

- In the packet header, the 'MSG' message is followed by the contact user's email address and chat-alias, followed by a number (during implementation, it was discovered that this number defines the number of characters in the packet)

- In the packet contents, fields b) and c) change, depending on whether the contact user is typing or sending a message, similarly to the behaviour of local user conversation packets.

- If the contact user is typing a message, we get a packet with an ending sequence of 3 newlines, where as in a message packet, we get a sequence of 2 newlines, followed by the chat message.

B.4 MSN Messenger Exit Chat Sequence

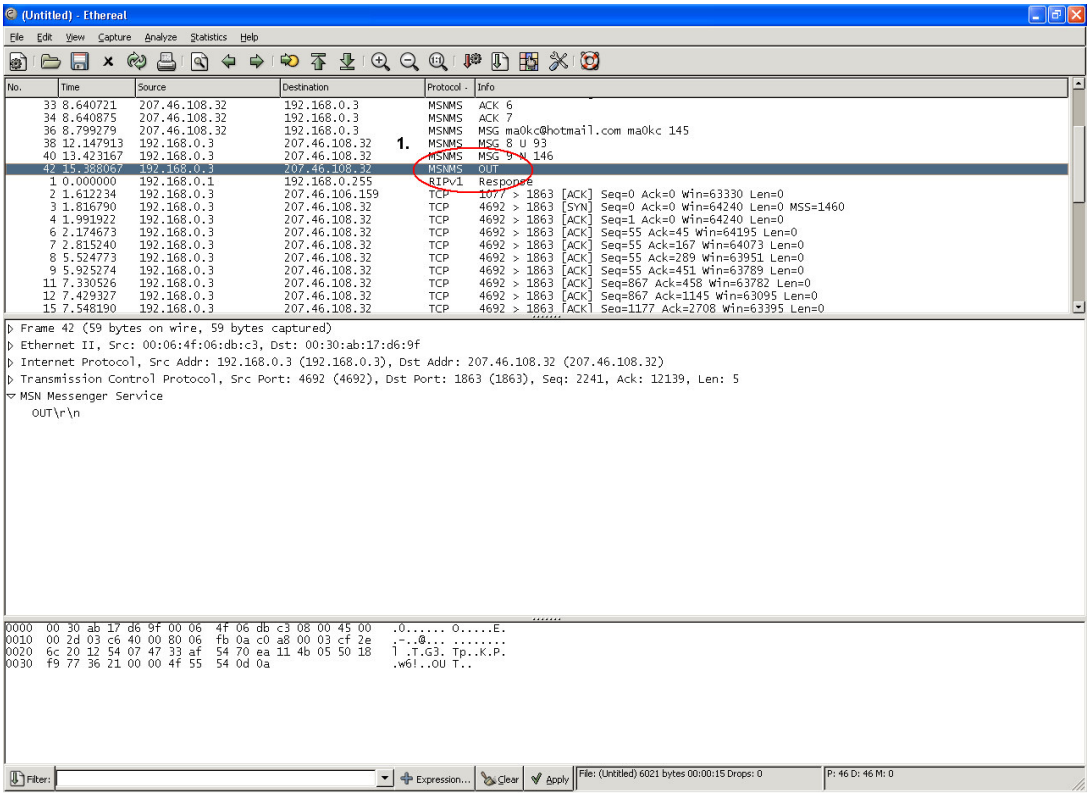

Figure B.4.1 Exit Chat Sequence

The procedure to exit a chat conversation is trivial, and defined as follows:

1. When a user leaves a chat conversation, we simply look out for the ‘OUT’ message.

The ‘OUT’ message follows the local user closing his/her chat window.

Appendix C: Serialization Code

SecureNetList

SecureNetList is a serialised template for the predefined categories

```
using System.Collections;
namespace SecureNet.Config
{       /// Serializable library class to store SecureNet IPBlockList information.
        [Serializable]
        public class SecureNetList
        {
                //WebSettings IPBlockLists
                public ArrayList AdultMatureList;
                public ArrayList DrugsList;
                public ArrayList EducationalList;
                public ArrayList EmailList;
                public ArrayList FinancialList;
                public ArrayList GamblingList;
                public ArrayList GamesList;
                public ArrayList KidsList;
                public ArrayList MusicList;
                public ArrayList NewsList;
                public ArrayList PoliticsList;
                public ArrayList ScienceList;

                //ChatSettingsIPBlockLists
                public ArrayList AOLChatList;
                public ArrayList ExciteChatList;
                public ArrayList IRCChatList;
                public ArrayList ICQChatList;
                public ArrayList MSNChatList;
                public ArrayList YahooChatList;

                public SecureNetList()
                {
                        AdultMatureList = new ArrayList();
                        DrugsList = new ArrayList();
                        EducationalList = new ArrayList();
                        EmailList = new ArrayList();
                        FinancialList = new ArrayList();
                        GamblingList = new ArrayList();
                        GamesList = new ArrayList();
                        KidsList = new ArrayList();
                        MusicList = new ArrayList();
                        NewsList = new ArrayList();
                        PoliticsList = new ArrayList();
                        ScienceList = new ArrayList();
                        AOLChatList = new ArrayList();
                        ExciteChatList = new ArrayList();
                        IRCChatList = new ArrayList();
                        ICQChatList = new ArrayList();
                        MSNChatList = new ArrayList();
                        YahooChatList = new ArrayList();
                }
        }
}
```

InitialiseSecureNetList

Initialises the list of predefined websites for each category

```
using System;
using System.Collections;
using System.Collections.Specialized;
using System.IO;
using System.Net;
using System.Net.Sockets;
using System.Runtime.Serialization;
using System.Runtime.Serialization.Formatters.Binary;
using SecureNet.Config;

/// <summary>
/// This class provides an example default categories list
/// </summary>
public class InitialiseSecureNetLists
{
	public static void Main()
	{
		InitialiseSecureNetLists list = new InitialiseSecureNetLists();
	}

	public InitialiseSecureNetLists()
	{

		SecureNetList list = new SecureNetList();

		//courtesy of typing 'porn' into google
		string[] AdultMatureList = {

"www.penisbot.com", "www.megapornlinks.com", "www.wowtgp.com",
"www.hereistheporn.com",    "www.persiankitty.com", "www.1freepornfinder.com",
"www.wetplace.com", "www.hardcore-hotel.org", “www.hardcorexxx.com”,
“www.XXXonline.co.uk”, “www.theteenhunt.com”, “www.sex.com",
"www.masterbates.com”, “www.pichunter.com”, “www.jerk-off-tgp.com”,
“www.xxxchurch.com", "www.porn-station.com", "www.freedailyporn.com",
"www.stickyhole.com", "xxx.adultsonlynet.com",   "www.adultcinema.org”, “www.southern-
charms.com", "www.penisdance.com”, “www.lastlonger.5u.com",    "www.fast-porn.net",
"www.sexzool.com", "www.porn-navigator.com”, “www.eroticy.com",   "www.search-4-
porn.com", "www.freepornlist.com”, “www.Quality-Adult-Sites.com”, “www.smsaxxess.co.uk",
"www.adult-list.com", "www.freehotgallery.com", "www.welovefreeporn.com" ,
"www.sexcyberbabes.com”, “www.xxxaccess.com",  “xxx.adultsonlynet.com",
"www.hotmoviesxxx.com”, “www.sex.com",  "www.hardcorejunky.com",
"www.ultrahardcoremovies.com”, “www.xxx-database.com”, “www.sexfireballs.com",
 "www.pornqueens.net”, “www.premiumxxxcore.com”, “www.sexynexus.com”,
“www.stileporn.com",  "www.porn-hitz.com”, “www.freepornlist.com”, “www.sex-xxx-porno-
pics.com”, “www.marissas.com",   "www.freesexbomb.com”, “www.freesitexxx.com”,
“www.getfreexxx.com”, “www.sex4kings.com"

		};

		//courtesy of typing 'drugs' into google
		string[] DrugsList = {


"www.drugs.com”, “www.clubdrugs.org”, “www.talktofrank.com”, “www.health.org",
 "www.stopdrugs.org”, “www.drugs.gov.uk”, “www.freevibe.com”, “www.dare.com",
 "www.drugs-forum.com”, “www.kidshealth.org”, “www.dancesafe.org”, “www.addictions.org",
```

"www.talkingwithkids.org/drugs.html “, “www.thegooddrugsguide.com”,
“www.drugsinsport.net”, “www.drugs.health.gov.au", "www.raceagainstdrugs.org”,
“www.drugsinfofile.com”, “www.drugs.greenparty.org.uk”, “www.thesite.org"

};

//courtesy of typing 'kids learn online' into google
string[] EducationalList = {

"www.funbrain.com”, “www.innerbody.com”, “www.uptoten.com”, “www.babloo.com",
"www.factmonster.com”, “www.geography4kids.com”, “pbskids.org”, “www.chesskids.com",
"www.nationalgeographic.com/kids”,”www.uptoten.com”,“www.hotwired.com/webmonkey/kids
”, “www.kids-online.net","www.prongo.com”,“www.syvum.com”,
“www.enchantedlearning.com”, “www.nickjr.com","kidshub.org”, “www.primarygames.com”,
“www.gamequarium.com”, “www.safekids.com", "www.kidsource.com”, “www.unac.org/learn”,
“www.abcteach.com”, “www.toonuniversity.com", "www.littleplanet.com”,
“www.cybersmartkids.com.au”, “www.kidsofamerica.com”, “www.gigglepotz.com"

};
//some well known web-based email providers
string[] EmailList = {

"www.hotmail.com”, “mail.yahoo.com”, “mail.yahoo.ca”, “mail.yahoo.co.jp",
"mail.yahoo.com.au”, “www.mail.com”, “www.myownemail.com”, “mail.lycos.co.jp",
"login.passport.net”, “mail.lycos.co.uk”, “mail.lycos.com”, “www.rock.com",
"gmail.google.com”, “www.everyone.net”, “www.freecenter.com/email.html”, “TuffMail.com",
"www.email2me.com”, “www.emailaccount.com/”, “www.hush.com”, “mailcircuit.com/"

};

//courtesy of typing 'finance' into google
string[] FinancialList = {

"finance.yahoo.com”, “www.cnnfn.com”, “www.fool.com”, “www.ifc.org",
"www.globeinvestor.com”, “www.quicken.com”, “www.finance-net.com”,
“www.financewise.com", "finance.lycos.com”, “www.financenter.com”, “www.kiplinger.com”,
“www.finweb.com", "moneymanager.smh.com.au”, “finance.pro2net.com”, “finance.wat.ch”,
“www.if.com", "www.ft.com”, “www.businessfinance.com”, “www.vfinance.com”,
“www.cipfa.org.uk"

};

//courtesy of typing 'gambling' into google
string[] GamblingList = {

"www.gambling.com”, “www.ncpgambling.org”, “www.gamblingtimes.com”, “www.gambling-
online-review.com","www.onlinegambling.com”, “www.dollarsgambling.com”,
“www.gamblingcasinoportal.com”, “www.gonegambling.com","www.ladbrokes.com”,
“www.bet365.com”, “www.gamble.co.uk”, “www.optimalgambling.com", "www.sports.com”,
“www.gambling-palace.com”, “www.gamblingtool.com”, “www.ukoga.org","www.gambling-
us.com”, “www.1classcasinos.com”, “www.gamblingproblem.org”, “gamblingcommission.com"

};

//courtesy of typing 'games' into google
string[] GamesList = {

"games.yahoo.com”, “www.funbrain.com”, “zone.msn.com”, “www.shockwave.com",

"www.games.com", "www.games-workshop.com", "www.gamespot.com", "www.gameskidsplay.net", "www.jumbo.com", "www.gamepro.com", "www.pogo.com", "www.miniclip.com", "www.rockstargames.com", "www.epicgames.com", "www.microsoft.com/games", "www.ea.com", "www.ign.com", "www.pcgamer.com", "www.freearcade.com", "www.ebgames.com"

};

//courtesy of typing 'kids' into google
string[] KidsList = {

"www.funschool.com", "www.billybear4kids.com", "www.primarygames.com", "www.kids-online.net","www.kidscom.com", "www.ajkids.com", "www.yahooligans.com", "kids.msfc.nasa.gov","www.kidsdomain.com", "www.factmonster.com", "www.kids-space.org", "kidshealth.org","kids.discovery.com", "www.fema.gov/kids", "www.eduplace.com/kids", "www.cyberkids.com"

};

//courtesy of typing 'music' into google
string[] MusicList = {

"www.mtv.com", "www.mp3.com", "www.allmusic.com", "launch.yahoo.com", "www.music.com", "www.cdnow.com", "www.musicmatch.com", "www.billboard.com", "www.apple.com/itunes", "www.vh1.com", "www.morpheus.com", "www.emusic.com", "www.napster.com", "www.sonymusic.com", "www.musicsearch.com", "www.classicalarchives.com", "www.musicnet.com", "music.lycos.com", "www.madonna.com", "www.grovemusic.com"

};

//courtesy of typing 'news' into google
string[] NewsList = {

"www.cnn.com", "news.bbc.co.uk", "abcnews.go.com", "www.foxnews.com", "news.google.com", "www.cbsnews.com", "www.wired.com", "www.news.com", "news.yahoo.com", "www.usnews.com", "www.reuters.com", "www.msnbc.com", "www.usatoday.com", "www.news.com.au", "www.theonion.com", "www.detnews.com", "www.cnet.com", "www.bloomberg.com"

};

//courtesy of typing 'politics' into google
string[] PoliticsList = {

"www.politics.com", "www.cnn.com/ALLPOLITICS", "www.nytimes.com/pages/politics", "www.opensecrets.org", "allpolitics.com", "www.politicsonline.com", "www.realclearpolitics.com", "www.cawp.rutgers.edu","www.jstor.org", "www.aboutpolitics.com", "www.lpig.org", "www.polisci.com","www.washingtonpost.com/wp-dyn/politics"

};

//courtesy of http://www.skewlsites.com - categorised educational sites
string[] ScienceList = {

"www.sciencespot.net", "www.dustbunny.com", "mpfwww.jpl.nasa.gov", "observe.arc.nasa.gov", "quest.arc.nasa.gov", "www.nyelabs.com", "www.sln.org", "chemistry.org", "library.thinkquest.org", "www.childrensmuseum.org",

“tqjunior.advanced.org”, “www.exploratorium.edu"

```
                };


                //ChatSettingsIPBlockLists
                string[] AOLChatList = { };
                string[] ExciteChatList = {};
                string[] IRCChatList = {};
                string[] ICQChatList = {};
                string[] MSNChatList = {};
                string[] YahooChatList = {};

                //resolve host names


                //insert ip addresses into SecureNetList
                list.AdultMatureList.AddRange(this.ResolveHosts(AdultMatureList));
                list.DrugsList.AddRange(this.ResolveHosts(DrugsList));
                list.EducationalList.AddRange(this.ResolveHosts(EducationalList));
                list.EmailList.AddRange(this.ResolveHosts(EmailList));
                list.FinancialList.AddRange(this.ResolveHosts(FinancialList));
                list.GamblingList.AddRange(this.ResolveHosts(GamblingList));
                list.GamesList.AddRange(this.ResolveHosts(GamesList));
                list.KidsList.AddRange(this.ResolveHosts(KidsList));
                list.MusicList.AddRange(this.ResolveHosts(MusicList));
                list.NewsList.AddRange(this.ResolveHosts(NewsList));
                list.PoliticsList.AddRange(this.ResolveHosts(PoliticsList));
                list.ScienceList.AddRange(this.ResolveHosts(ScienceList));
                list.AOLChatList.AddRange(this.ResolveHosts(AOLChatList));
                list.ExciteChatList.AddRange(this.ResolveHosts(ExciteChatList));
                list.IRCChatList.AddRange(this.ResolveHosts(IRCChatList));
                list.ICQChatList.AddRange(this.ResolveHosts(ICQChatList));
                list.MSNChatList.AddRange(this.ResolveHosts(MSNChatList));
                list.YahooChatList.AddRange(this.ResolveHosts(YahooChatList));

                Stream str = new FileStream("SCN.lst", FileMode.Create,
                FileAccess.ReadWrite);
                IFormatter formatter = new BinaryFormatter();

                formatter.Serialize(str, list);
                str.Close();

        }

        //perform a nslookup for each defined host
        private ArrayList ResolveHosts(string[] list)
        {
                ArrayList hosts = new ArrayList();

                foreach (string host in list)
                {
                        try
                        {
                                IPHostEntry results = Dns.GetHostByName(host);
                                foreach(IPAddress address in results.AddressList)
                                {
                                        hosts.Add(address);
                                }
                        }
```

```
                    catch {} //couldn't resolve name
                }
                return hosts;
            }
        }
```

Config

The configuration library responsible for holding configuration data

```
using System;
using System.Collections;

namespace SecureNet.Config
{
	/// <summary>
	/// Serializable library class to store SecureNet client configuration
	/// data elements.
	/// </summary>
	[Serializable]
	public class Config
	{
		/// <summary>
		/// SecureNetSettings Configurations
		/// </summary>
		public string SecureNetServerIP;
		public string SecureNetServerPort;
		public bool AllowRemoteDesktop;
		public bool ProxySet;
		public string ProxyIP;
		public string ProxyPort;
		public bool AutoDownLoadURLList;
		public int MaxWebLogSize;
		public int KeepWebLogsFor;
		public int UploadWebLogsFrequency; //0=immediately, 1=1hr, 2=2hrs,
3=everyday, 4=everyweek
		public int MaxChatLogSize;
		public int KeepChatLogsFor;
		public int UploadChatLogsFrequency; //0=immediately, 1=1hr, 2=2hrs,
3=everyday, 4=everyweek

		/// <summary>
		/// WebLogSettings Configurations
		/// </summary>
		/// <remarks>0=blocked, 1=allowed</remarks>
		public int SetAdultMatureList;
		public int SetDrugsList;
		public int SetEducationalList;
		public int SetEmailList;
		public int SetFinancialList;
		public int SetGamblingList;
		public int SetGamesList;
		public int SetKidsList;
		public int SetMusicList;
		public int SetNewsList;
		public int SetPoliticsList;
		public int SetScienceList;
		public int WebLoggingLevel; // 0=no logging, 1=log blocked, 2=log everything

		//WebLogSettings_CustomAllowedList
		public ArrayList CustomAllowedList;

		//WebLogSettings_CustomBlockedList
		public ArrayList CustomBlockedList;
```

```
        /// <summary>
        /// ChatLogSettings Configurations
        /// </summary>
        public int AllowAOLChat;
        public int AllowExciteChat;
        public int AllowIRCChat;
        public int AllowICQChat;
        public int AllowMSNChat;
        public int AllowYahooChat;

        public ArrayList ProhibitedWordsList;
        public int ChatLoggingLevel;  //0=no logging, 1=log when prohibited content detected, 2=log everything
        public int ChatAction;  //0=do nothing, 1=remove&replace, 2=replace entire message
        public string CensorString;
        public string ReplacementMsgString;

        /// <summary>
        ///  UserSettings Configuration
        ///  Only used to store information about users' sensitive data
        /// </summary>
        public ArrayList SensitiveDataList;
        public int SecurityLevel; //0=allow everything, 1=allow everything except blocklist, 2=block everything except both allowlists, 3=allow custom allowlist items only

        /// <summary>
        /// Constructor
        /// </summary>
        public Config()
        {
            //SecureNet Settings
            SecureNetServerIP = null;
            SecureNetServerPort = null;
            AllowRemoteDesktop = true;
            ProxySet = false;
            ProxyIP = null;
            ProxyPort = null;
            AutoDownLoadURLList = false;
            MaxWebLogSize = 10;
            KeepWebLogsFor = 30;
            UploadWebLogsFrequency = 1; //0=immediately, 1=1hr, 2=2hrs, 3=everyday, 4=everyweek
            MaxChatLogSize = 10;
            KeepChatLogsFor = 30;
            UploadChatLogsFrequency = 0; //0=immediately, 1=1hr, 2=2hrs, 3=everyday, 4=everyweek

            //WebLogSettings
            SetAdultMatureList = 0;          //0=blocked, 1=allowed
            SetDrugsList = 0;
            SetEducationalList = 1;
            SetEmailList = 1;
            SetFinancialList = 1;
            SetGamblingList = 1;
            SetGamesList = 1;
            SetKidsList = 1;
            SetMusicList = 1;
            SetNewsList = 1;
            SetPoliticsList = 1;
```

```
                        SetScienceList = 1;
                        WebLoggingLevel = 1; // 0=no logging, 1=log blocked 2=log
everything

                        //WebLogSettings_CustomBlockedList
                        CustomAllowedList = new ArrayList();

                        //WebLogSettings_CustomBlockedList
                        CustomBlockedList = new ArrayList();

                        //ChatLogSettings Configurations
                        AllowAOLChat = 0;       //0 true, 1 false
                        AllowExciteChat = 0;
                        AllowIRCChat = 0;
                        AllowICQChat = 0;
                        AllowMSNChat = 0;
                        AllowYahooChat = 0;

                        ProhibitedWordsList = new ArrayList();
                        ChatLoggingLevel = 1;   //0=no logging, 1=log when prohibited
content detected, 2=log everything
                        ChatAction = 0;  //0=do nothing, 1=remove&replace, 2=replace entire
message
                        CensorString = "<censored>";
                        ReplacementMsgString = "The content of this message was deemed
inappropriate.";

                        //UserSettings Configuration
                        SensitiveDataList = new ArrayList();
                        SecurityLevel = 1;
                }
        }
}
```

SetReg

Responsible for installing SecureNet service and registry keys

```
using System;

/// <summary>
/// Setup file for SecureNet client - Initialises registry keys to setup proxy settings.
/// </summary>
public class Setup
{
	public Setup()
	{
		//set acceptable use agreement
		Microsoft.Win32.RegistryKey warning =
	Microsoft.Win32.Registry.LocalMachine.OpenSubKey("Software\\Microsoft\\Windows
	NT\\CurrentVersion\\WinLogon", true);
		warning.SetValue("forceunlocklogon", 1);
		warning.SetValue("LegalNoticeCaption", "Warning: This computer is being
monitored by SecureNet.");

		string text = @"By logging onto this computer I agree to the following:

- I will not give out personal information such as my address, telephone number, parents'
work address/telephone number, or the name and location of my school without my parents'
permission.

- I will never agree to get together with someone I meet online without first checking with my
parents. If my parents agree to the meeting, I will be sure that it is in a public place and bring
my mother or father along.

- I will tell my parents right away if I come across any information that makes me feel
uncomfortable.

- I will not respond to any messages that are mean or in any way make me feel
uncomfortable. It is not my fault if I get a message like that. If I do I will tell my parents right
away so that they can contact the online service.

Failure to comply with these rules may lead to disciplinary action and the withdrawal of
computer privileges.";

		warning.SetValue("LegalNoticeText", text);

		//set proxy settings
		//Internet Explorer
		Microsoft.Win32.RegistryKey ieproxyset =
Microsoft.Win32.Registry.CurrentUser.OpenSubKey("Software\\Microsoft\\Windows\\CurrentV
ersion\\Internet Settings", true);
		ieproxyset.SetValue("ProxyServer", "127.0.0.1:80");
		ieproxyset.SetValue("ProxyEnable", 1);

		//Messenger
		Microsoft.Win32.RegistryKey msnproxyset =
Microsoft.Win32.Registry.CurrentUser.OpenSubKey("Software\\Microsoft\\MSNMessenger",
true);
		msnproxyset.SetValue("SOCKS4Server", "127.0.0.1");
		//byte[] port = {38, 04, 00, 00};

		byte[] port = {(byte)56,(byte)4, (byte)0, (byte)0};
```

```
                msnproxyset.SetValue("SOCKS4Port", port);
                byte[] enable = {(byte)255,(byte)255,(byte)255,(byte)255 };
                msnproxyset.SetValue("ProxyState", enable);

        }

        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        public static void Main(string[] args)
        {
                //
                // TODO: Add code to start application here
                //
                Setup setup = new Setup();
        }
}
```

ConvoCapture

Responsible for keeping conversation log data

```
using System;
using System.Collections;

namespace SecureNet.SCNCapture
{
	/// <summary>
	/// Serializable class to store captured conversation
	/// data elements.
	/// </summary>
	[Serializable]
	public class ConvoCapture
	{
		public string User;
		public string LocalEmail;
		public string ContactEmail;
		public string ContactAlias;
		public string Time;
		public bool InappropriateContent;
		public bool SensitiveDataRelease;
		public ArrayList Messages;
		public ArrayList ViolationKeys;

		public ConvoCapture()
		{
			User = "";
			LocalEmail = "";
			ContactEmail = "";
			ContactAlias = "";
			Time = "";
			InappropriateContent = false;
			SensitiveDataRelease = false;
			Messages = new ArrayList();
			ViolationKeys = new ArrayList();

		}
	}
}
```

URLCapture

Responsible for keeping conversation log data

```
using System;

namespace SecureNet.SCNCapture
{
	/// <summary>
	/// Serializable class to store web access details.
	/// </summary>
	[Serializable]
	public class URLCapture
	{
		public string User;
		public string URL;
		public string IP;
		public string Action;
		public string Time;

		public URLCapture()
		{
			User = "";
			URL = "";
			IP = "";
			Action = "";
			Time = "";
		}
	}
}
```

Appendix D: Mentalis Proxy Overview

Proxy is a C# implementation of an Http, Ftp and Socks proxy server and PortMap server, developed and distributed as open source by Mentalis. All the classes are fully documented and as such it was possible to rip apart the application and use the Http and Socks engine for the SecureNet packet analyser.

The Proxy application consists of several main classes which we examined:

Proxy Class

The proxy class is the main driver for the proxy application, responsible for controlling the settings and listener objects, and displaying the console application seen below:

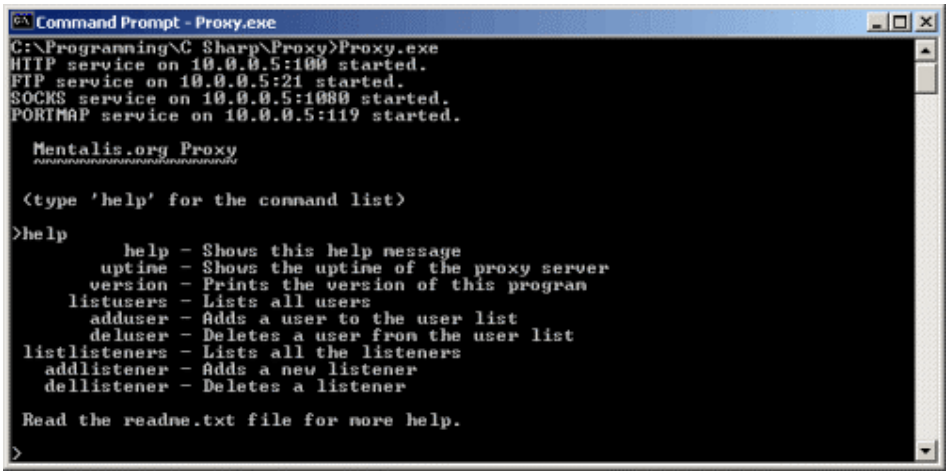

Figure C.1 Mentalis Proxy Console

On start-up it will attempt to look for a previously defined configuration file, otherwise the user has to define a proxy server-type and define the IP address and port to listen on.

Config Class

The config class is responsible for loading and saving the configuration file defined by the user using the console within the Proxy class. The configuration is held within an XML file, listing each type of proxy server required, and the port it is listening on.

Listener Class

The listener class specifies the basic methods and properties of a Listener object. The Listener class provides an abstract base class that represents a listening socket of the proxy server. Descendant classes further specify the protocol that is used between those two connections.

Client Class

The Client class provides an abstract base class that represents a connection to a local client and a remote server. Descendant classes further specify the protocol that is used between those two connections.

Http

Implementing a Http proxy involves using the HttpListener class, which listens on a specific port on the proxy server, defined by a user in the console window. When a Http request is received, it calls the HttpClient class which forwards all incoming Http traffic to the appropriate server. If a connection is established with the remote server, HttpClient is then responsible for relaying Http data between the remote host and a local client

Socks

Implementing a Socks proxy involves using the SocksListener class, which listens on a specific port on the proxy server for incoming Socks4 and Socks5 requests. If a Socks request is received, SocksHandler deals with the Socks negotiation. When complete, SocksClient is then responsible for relaying data between the remote server and a local client.

Appendix E: SecureNet Screen Design

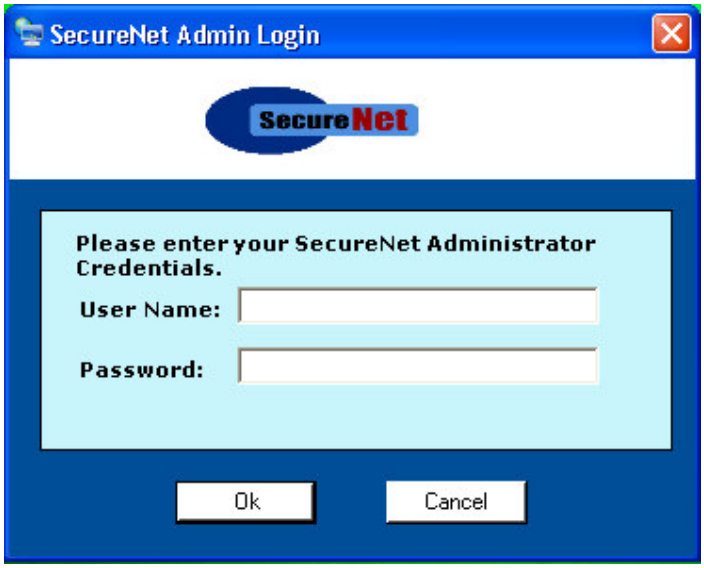

Figure E.1 SecureNet Admin, Logon Screen

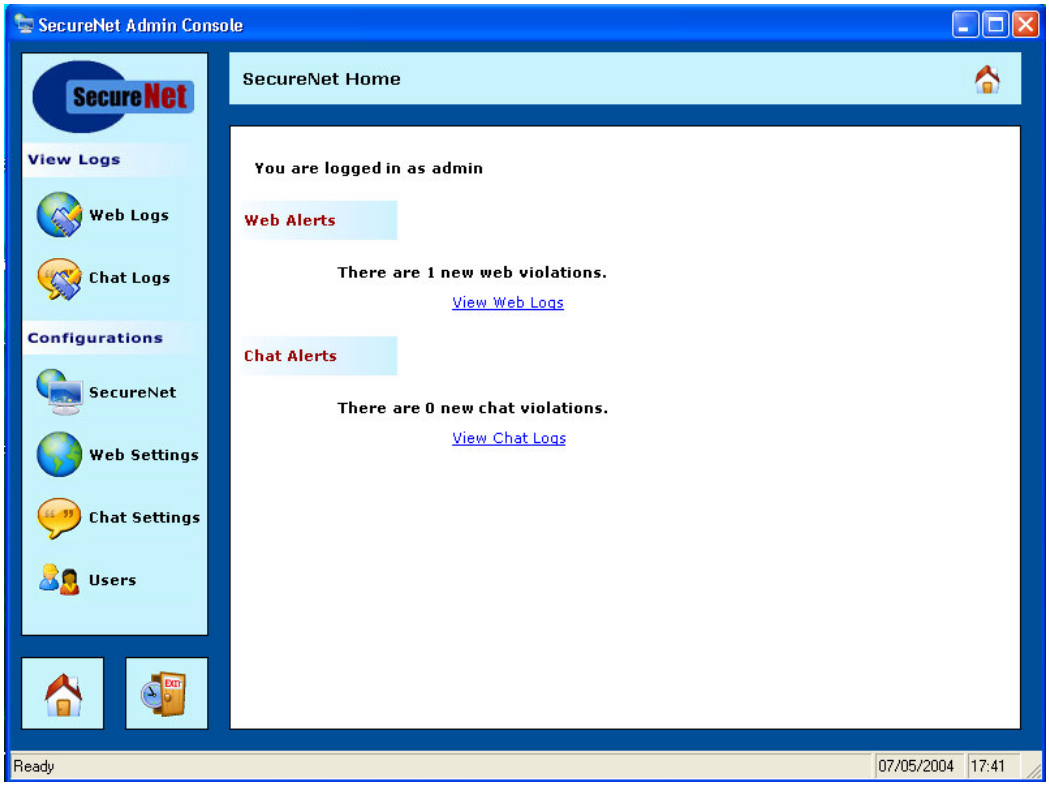

Figure E.2 SecureNet Admin, Home Screen

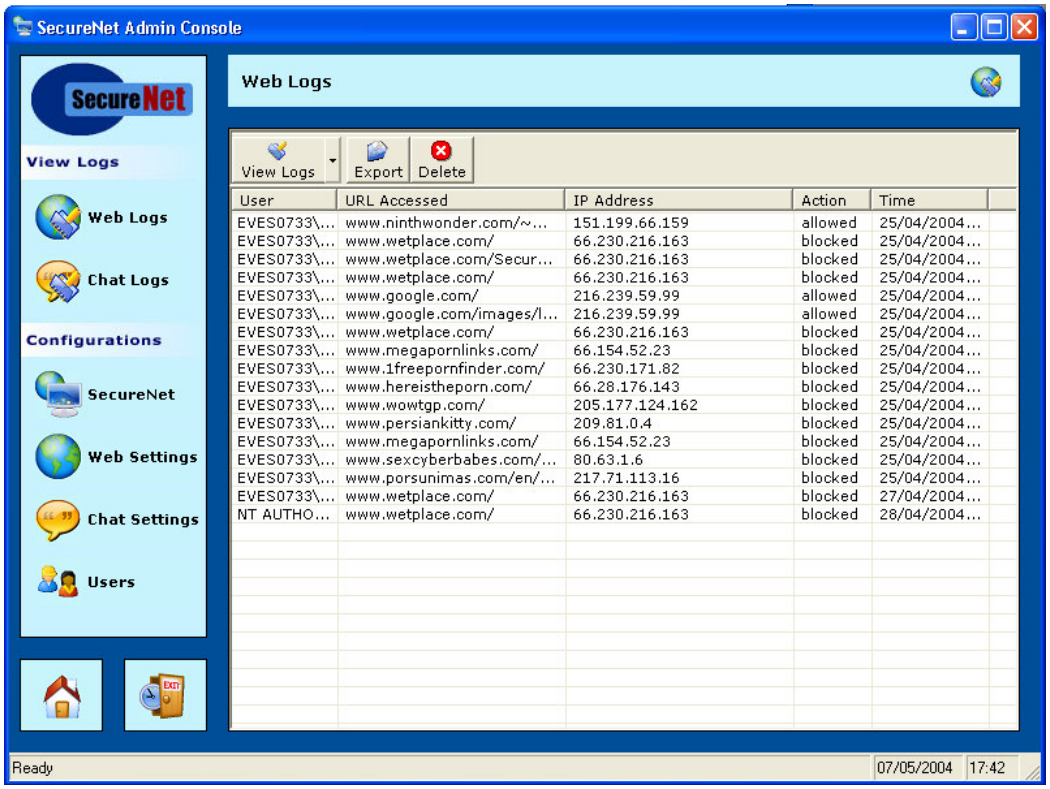

Figure E.3 SecureNet Admin, Web Logs Screen

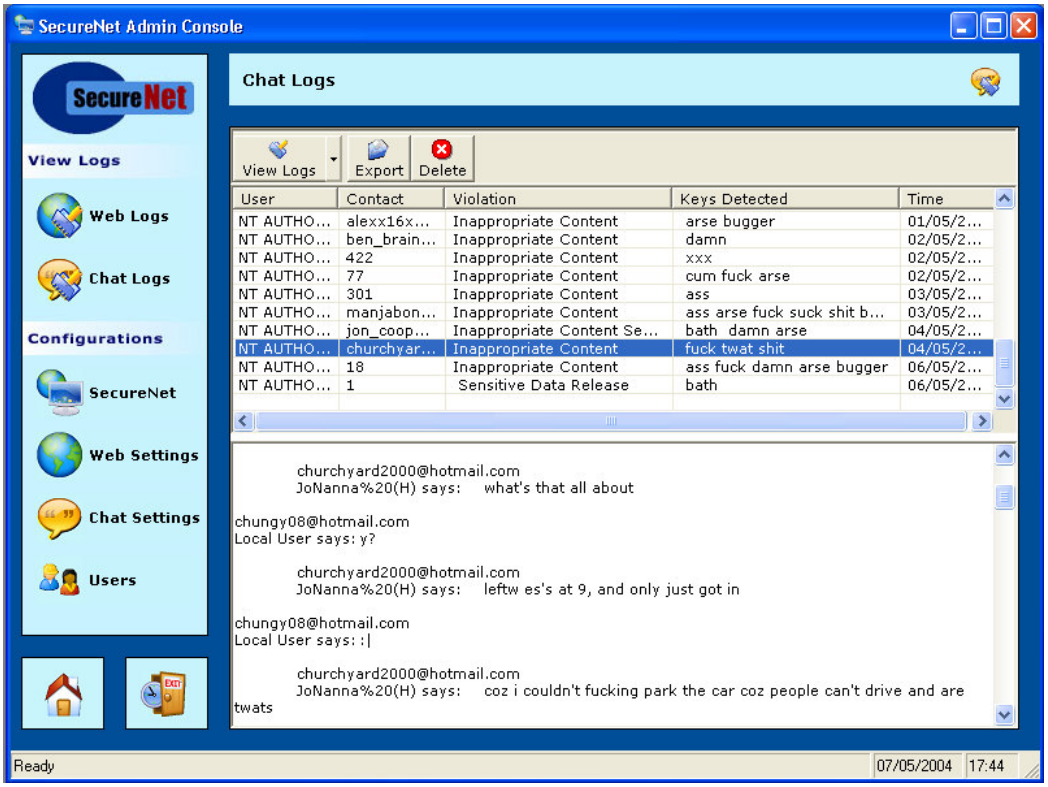

Figure E.4 SecureNet Admin, Chat Logs Screen

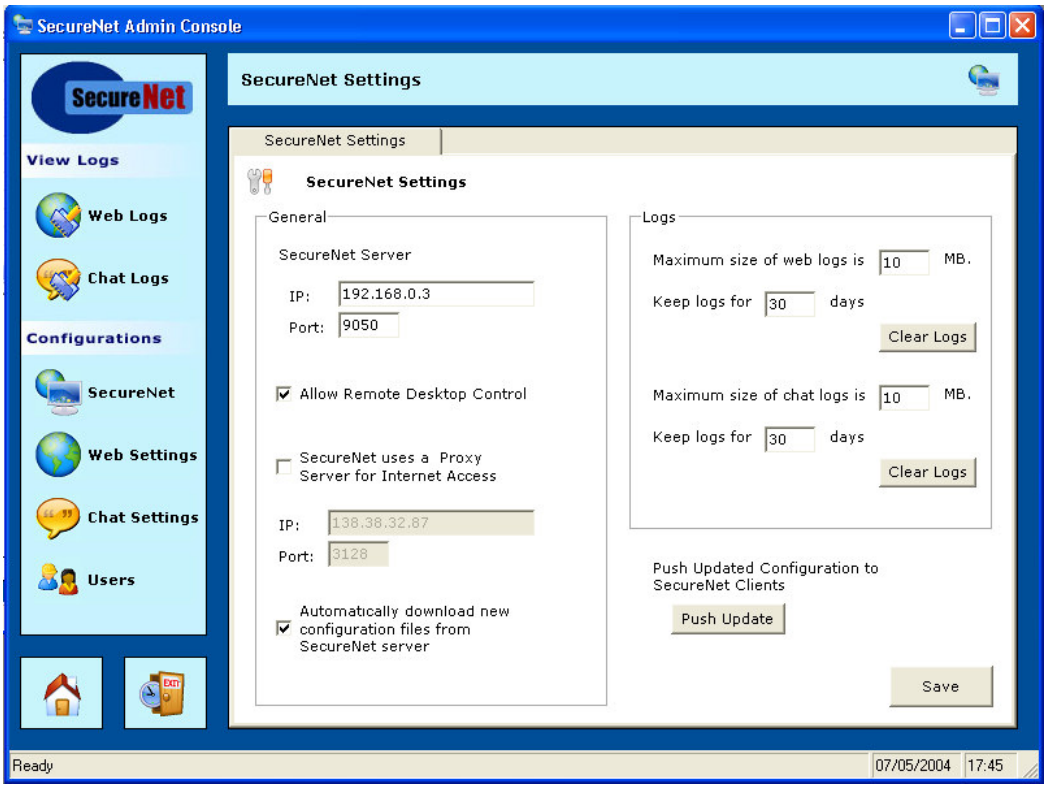

Figure E.5 SecureNet Admin, SecureNet Settings Screen

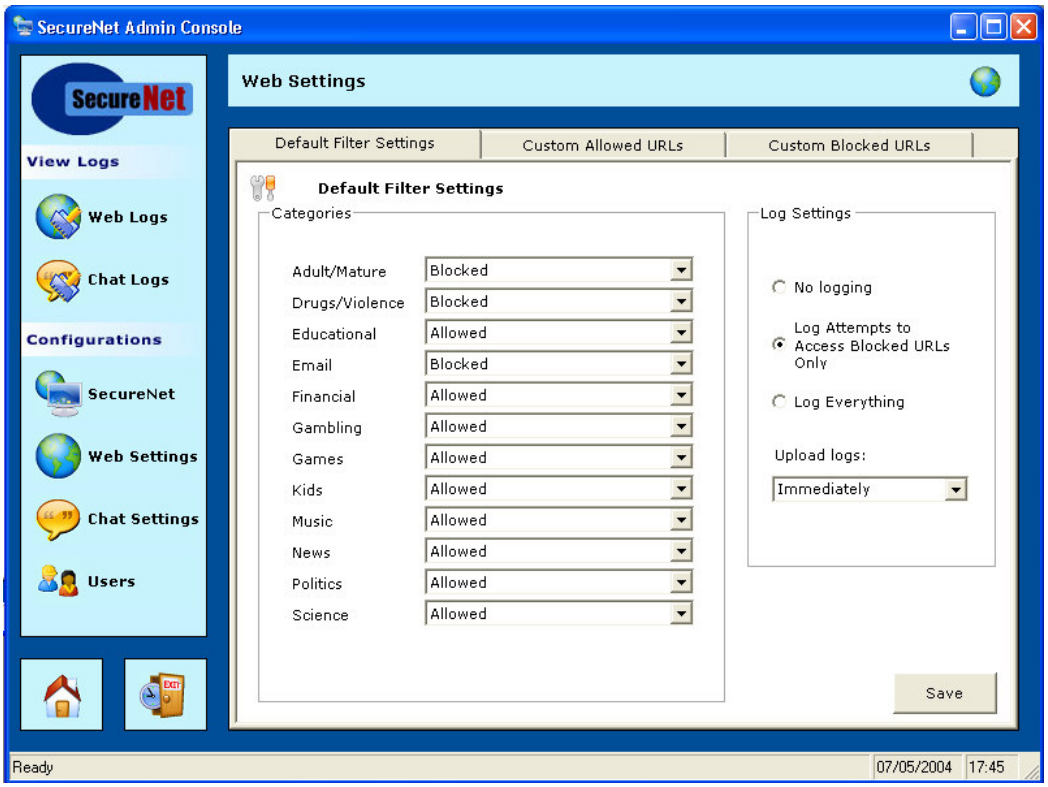

Figure E.6 SecureNet Admin, Web Settings 'Default Filter' Screen

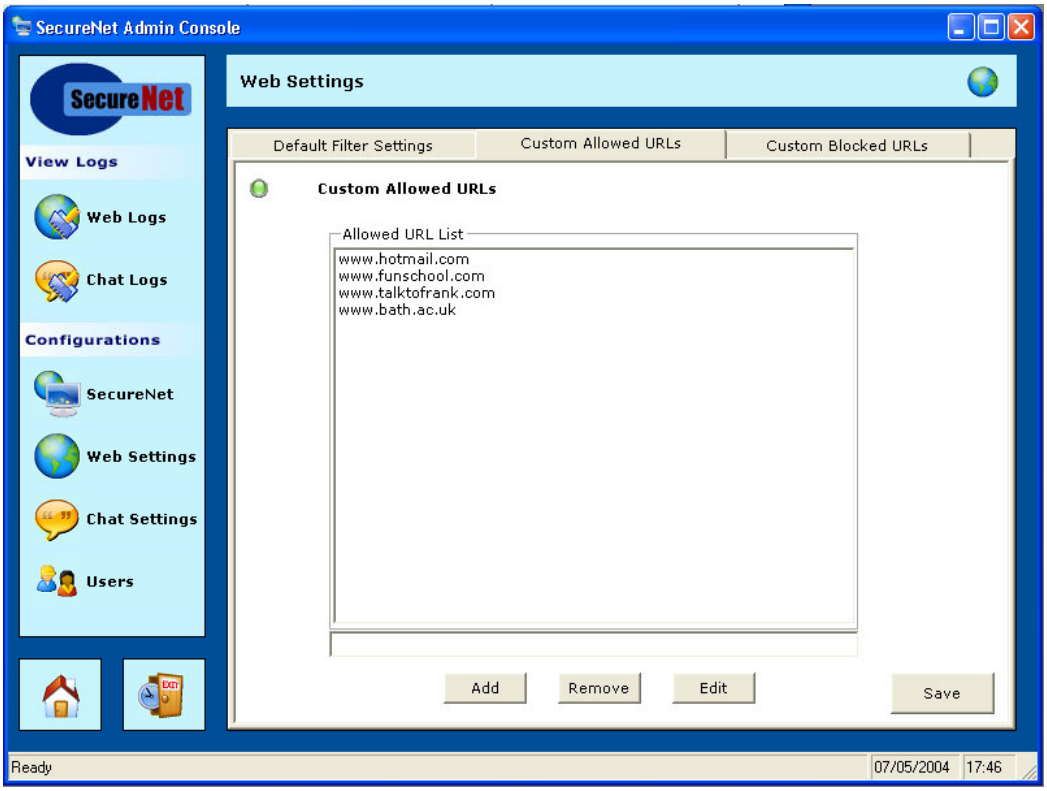

Figure E.7 SecureNet Admin, Web Settings 'Custom Allowed' Screen

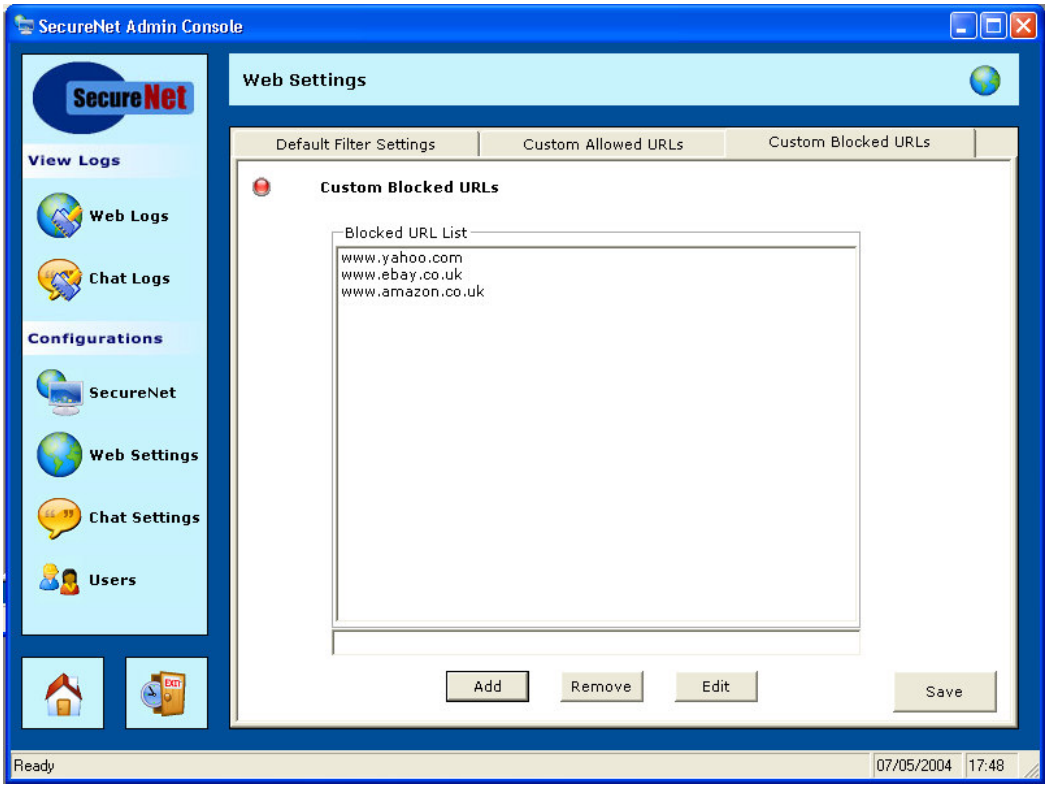

Figure E.8 SecureNet Admin, Web Settings 'Custom Blocked' Screen

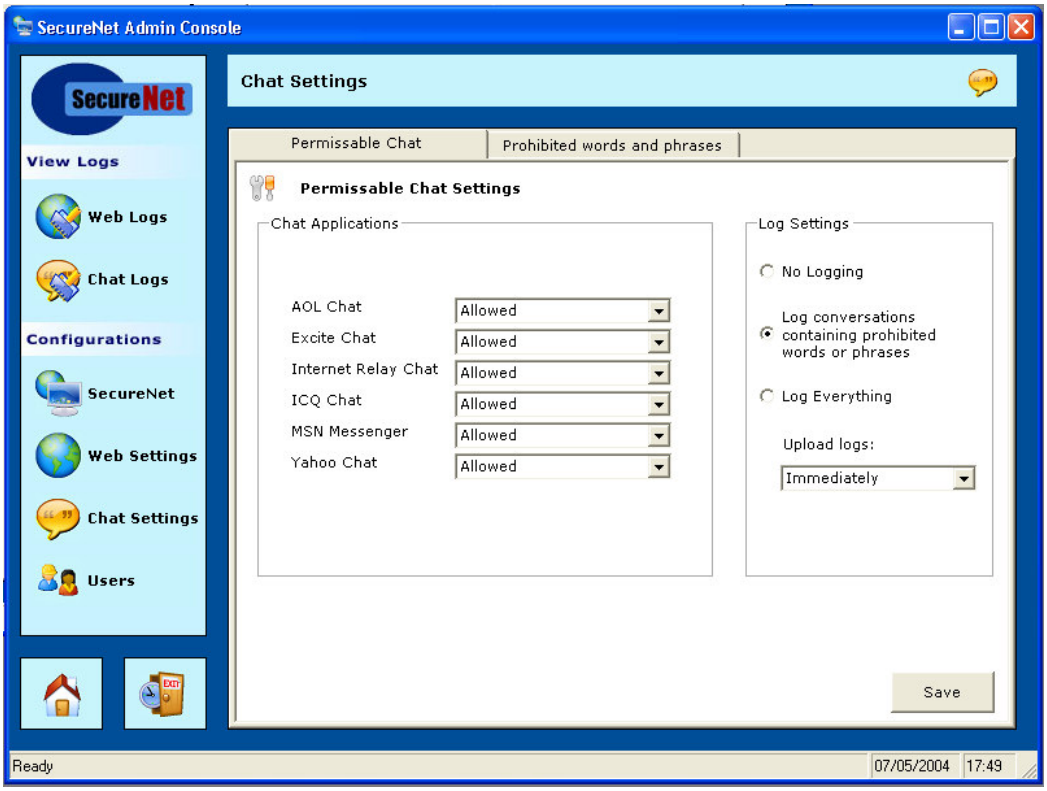

Figure E.9 SecureNet Admin, Chat Settings 'Permissible Chat Programs' Screen

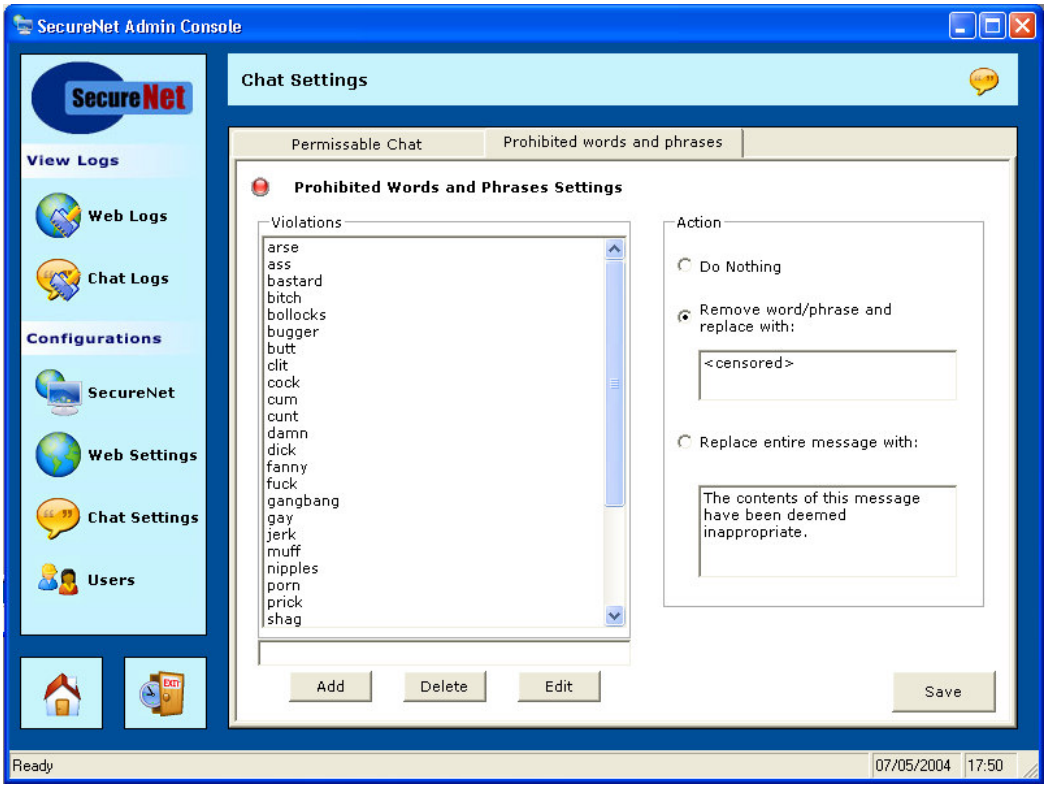

Figure E.10 SecureNet Admin, Chat Settings 'Prohibited Words & Phrases' Screen

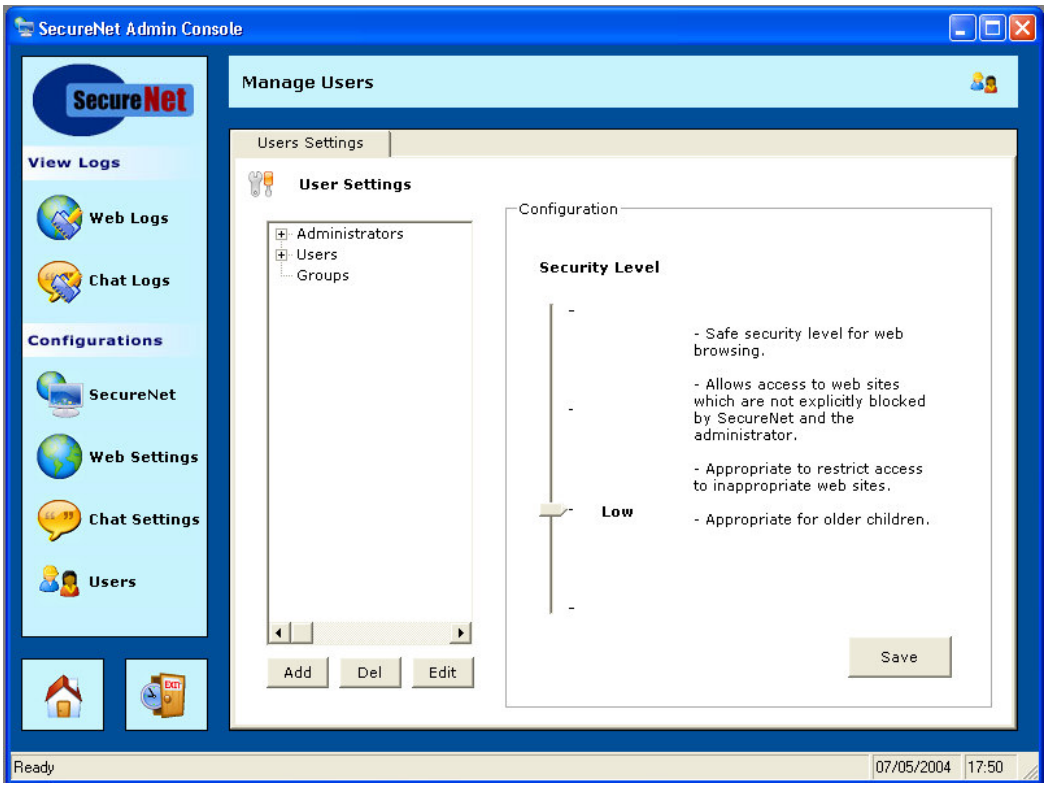

Figure E.11 SecureNet Admin, User Settings 'Security Level' Screen

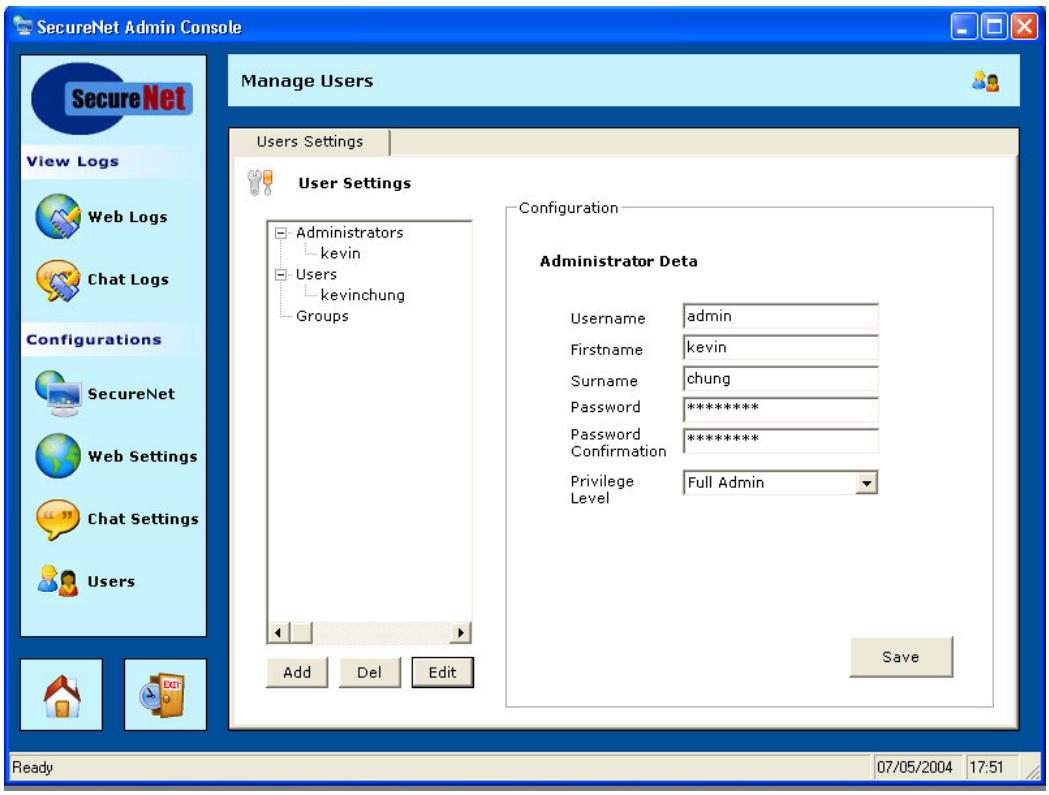

Figure E.12 SecureNet Admin, User Settings 'Admin Configuration' Screen

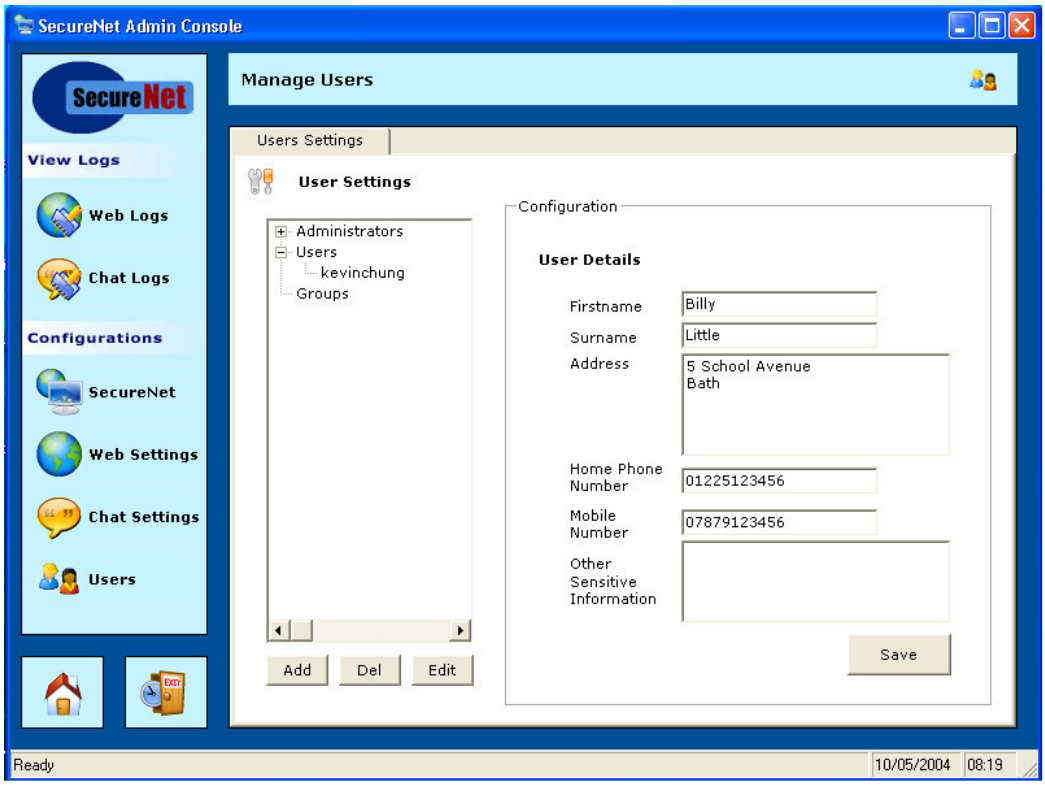

Figure E.13 SecureNet Admin, User Settings 'User Configuration' Screen

Figure E.14 SecureNet Admin, System Tray Function

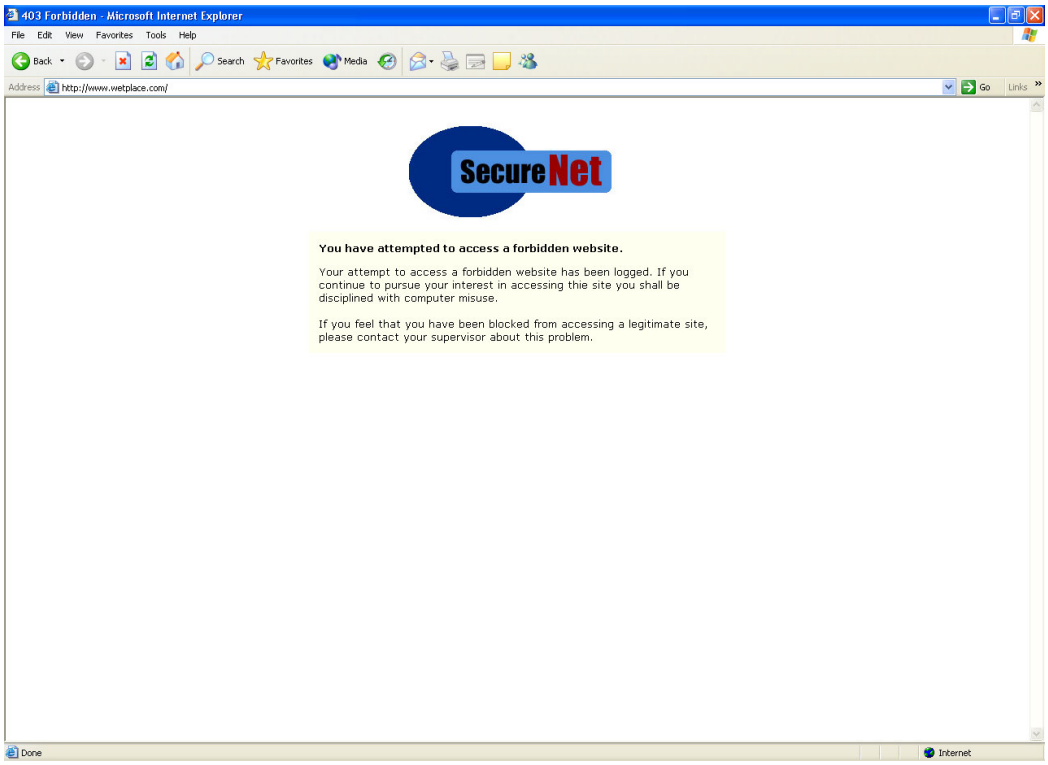

Figure E.15 SecureNet 'Blocked URL' Screen

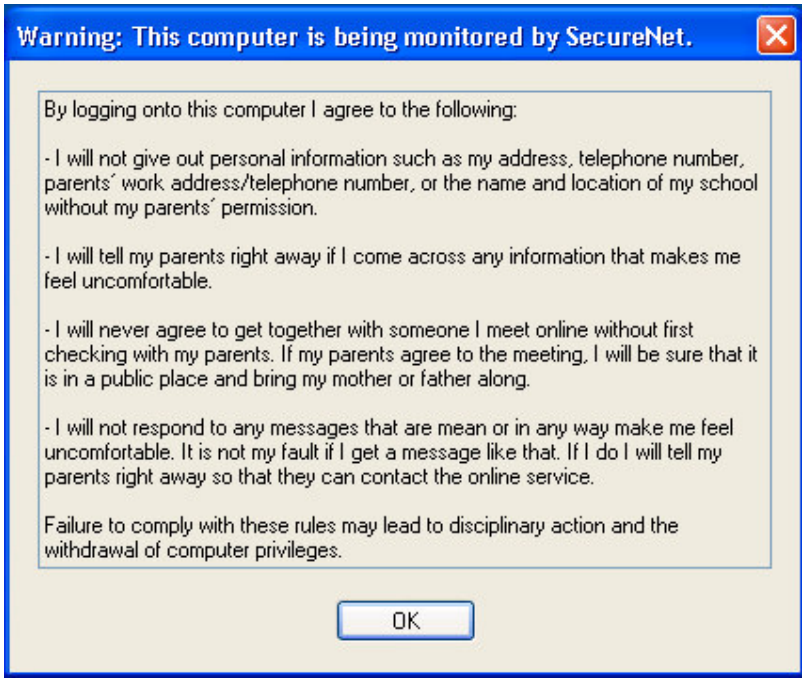

Figure E.16 Warning Message